APPLICATIONS OF CODING THEORY TO SUB-LINEAR TIME SPARSE RECOVERY PROBLEMS

A Dissertation

by

NAGARAJ THENKARAI JANAKIRAMAN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Krishna R. Narayanan
Committee Members,	Jean-Francois Chamberland-Tremblay
	Anxiao (Andrew) Jiang
	Alex Sprintson
Head of Department,	Miroslav M. Begovic

December 2020

Major Subject: Electrical and Computer Engineering

ABSTRACT

This dissertation leverages connection between coding theory and classical sparse recovery problems like sparse Fourier and Hadamard transform computations to understand properties of existing recovery algorithms under various signal models, propose improvements, and adopt them to interesting applications in theoretical computer science like pattern matching.

In the first part of the dissertation, we begin by demonstrating the relationship between an extended Fast Fourier Aliasing-based Sparse Transform (FFAST) algorithm and the iterative hard decision decoding of product codes. We show that the FFAST algorithm is analogous to an iterative decoder for a carefully defined product code whose thresholds can be computed by an extension of Justensen [1] analysis to d-dimensional product codes. Interpreting the FFAST algorithm as decoding of a product code also provides insight into the performance of the FFAST algorithm when non-zero coefficients are not randomly chosen, but are bursty such as what may be encountered in many practical applications like spectrum sensing. Recoverability results are guaranteed for the finite length case and we provided thresholds for the 1 and 2 burst cases asymptotically. It is further observed that the FFAST algorithm performs better for bursty signals in comparison to those for randomly chosen non-zero coefficients.

We then consider the problem of computing the Walsh-Hadamard Transform (WHT) of an $N = 2^n$-dimensional signal whose WHT is K-sparse, when the sparsity parameter $K = O(N^\delta)$ scales sub-linearly in N for some $0 < \delta < 1$. We propose improvements to the algorithm in [2] by introducing a two error correcting code at each check node. Further, through density evolution analysis and simulations

we show that the proposed modification substantially improves the space and time complexity of the algorithm, sometimes achieving as much as a 70% reduction.

We conclude by considering the substring/pattern matching problem of querying a string (or a database) of length N bits to determine all the locations where a substring (query) of length M appears either exactly or is within a Hamming distance of K from the query. We analyze the exact pattern matching problem where M consecutive symbols from $\vec{x}$ and is presented as a query, and the approximate pattern matching problem where we assume a noisy version of a substring. The algorithm finds application in two scenarios- when the string $\vec{x}$ is available before querying and one time computations can be performed offline and stored or when the string $\vec{x}$ is not centrally available, but parts of the string are sensed by different data collecting nodes distributively and communicated to a central server. Our proposed algorithm is evaluated based on the sketching complexity, and the computational complexity in answering the query. Using a sparse Fourier transform computation based approach we show that all such matches can be determined with high probability in sub-linear time and space. Further, we present several extensions including optimization for longer query lengths, algorithmic improvements for correlated data sources, and a secured matching algorithm in an outsourced pattern matching setting.

DEDICATION

To my parents and grandparents.

ACKNOWLEDGMENTS

It has been a long, exciting and satisfying path to my doctoral degree. If not for the love and motivation of many people, beginning from my undergraduate teachers and classmates to my professors, friends and peers at Texas A&M University, this would not have been achievable. My sincere thanks and appreciation go out to everyone who supported me.

First of all, I would like to express my heartfelt gratitude to my advisor Dr. Krishna Narayanan for his support during my graduate program. This dissertation cannot have been accomplished without his consistent support. I will be forever grateful to his teachings, encouragement and valuable guidance in times of hardship. In my first semester at Texas A&M University, he taught me digital signal processing and his passionate teaching made me develop interest for the subject and laid the groundwork for this dissertation. I am blessed to have spent my academic career's formative years under his guidance. I would like to thank professors Jean-Francois Chamberland, Andrew Jiang, Sivakumar Natarajan and Alex Sprintson for their valuable time in serving on my thesis committee. I am grateful for the perspectives they gave.

I am grateful to many friends and colleagues who made this experience much more fun. I'd like to note and thank my roommates Manoj, Bhaskar, Bhagyaraja, Puneet, Sreeram, Abhay, Ashok and Aadil for all the wonderful times we had at grad school. I am also thankful to my lab-mates Avinash, Santhosh, Narayanan, Arman, Vamsi, Ying, Adarsh, and several others for all the insightful discussions.

Above all, I am immensely thankful to Srividhya Balaji and Ashwin Krishna for

all the great moments we had at College Station, for being there for me in the good and bad times of this ride, for providing constant care, inspiration, encouragement, advice and support at all times.

Lastly, I would like to dedicate this work to my parents, Janakiraman and Kamakshi, and to my brother, Shankar, for their endless support, encouragement, and for the tremendous confidence they have in me. I'm indebted to my mom for the endless sacrifices she has made for us, and my father for being our role model, and insisting that my brother and I value education over everything else. I'll be grateful to them forever.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Krishna Narayanan (advisor) and Professors Jean-Francois Chamberland and Alex Sprintson of the Department of Electrical and Computer Engineering, Texas A&M University, Professor Andrew Jiang of the Department of Computer Science, Texas A&M University and Professor Sivakumar Natarajan of the Department of Mathematics, Texas A&M University. The work presented in Chapter 3 is carried out in collaboration with Professor Kannan Ramchandran of the Department of Electrical Engineering and Computer Science, University of California, Berkeley and Santhosh Emmadi, a graduate student in the Department of Electrical and Computer Engineering, Texas A&M University. The work presented in Chapter 4 is carried out in collaboration with Professor Robert Calderbank of the Department of Computer Science, Electrical Engineering, and Mathematics, Duke University. The work presented in Chapter 5 was carried out in collaboration with Avinash Vem, a graduate student in the Department of Electrical and Computer Engineering, Texas A&M University. All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and a research grant from the National Science Foundation.

NOMENCLATURE

BWT	Burrows-Wheeler Transform
DE	Density Evolution
DFT	Discrete Fourier Transform
FFAST	Fast Fourier Aliasing-based Sparse Transform
FFT	Fast Fourier Transform
IDFT	Inverse Fourier Transform
RS	Reed-Solomon
RSIDFT	Robust Sparse Inverse Discrete Fourier Transform
SNR	Signal to Noise Ratio
WHT	Walsh Hadamard Transform

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Sparse recovery, or compressed sensing, is an important mathematical tool that aims at recovering a K-sparse hidden vector $\vec{x} \in \mathbb{R}^N$ using $M < N$ linear measurements of the form $\vec{y} = \mathbf{A}\vec{x}$, derived from a sensing matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$. The measurement vector $\vec{y}$ is also known as the sketch of $\vec{x}$. The primary aim here is to minimize the number of measurements M used to represent $\vec{x}$, and the complexity of the recovery algorithm to reconstruct $\vec{x}$ from $\vec{y}$. Sparse recovery algorithms find applications in numerous domains such as compression, data stream computing [4, 5, 6], group testing [7, 8, 9], image processing [10, 11, 12], pattern matching [13] etc. With the exponential increase of data in the 21st century, algorithms with linear processing time proves infeasible to meet the demand. Therefore, it is critical to design algorithms with sub-linear time complexity.

This dissertation focuses on three sparse recovery problems i) computing a sparse Fourier transform, ii) computing a sparse Hadamard transform and iii) pattern matching. We leverage connections between coding theory and sub-linear time sparse recovery problems like sparse Fourier and Hadamard transform computations. This helps us understand properties of existing recovery algorithms under various signal models and propose algorithmic improvements to reduce space and time complexity. We finally consider an interesting application in theoretical computer science called pattern matching and present a sub-linear space and time algorithm to recover the matches with high probability. An outline of the dissertation is presented below.

1.1 Background

In Chapter 2, we review some fundamentals of signal processing, sparse recovery and coding theory concepts that will be used throughout the dissertation. The chap-

ter begins by briefly reviewing fundamentals of Fourier and Hadamard transform. This is followed by a brief review of the Fast Fourier Aliasing based Sparse Transform (FFAST) algorithm, a sparse recovery algorithm that was proposed by Pawar and Ramachandran in [3].

1.2 Sparse Fourier Transform

In Chapter 3, we consider the problem of computing sparse Discrete Fourier Trasnsform (DFT) of a N dimensional signal whose DFT is K-sparse. We show that the Fast Fourier Aliasing-based Sparse Transform (FFAST) algorithm for computing the Discrete Fourier Transform (DFT) [3] of signals with a sparse DFT is equivalent to iterative hard decision decoding of product codes. This connection is used to derive the thresholds for sparse recovery based on a recent analysis by Justensen [1] for computing thresholds for product codes. We first extend Justesen's analysis to d-dimensional product codes and compute thresholds for the FFAST algorithm based on this. Additionally, this connection also allows us to analyze the performance of the FFAST algorithm under a burst sparsity model in addition to the uniformly random sparsity model which was assumed in prior work [3].

1.3 Sparse Walsh Hadamard Transform

Next in Chapter 4, we consider the problem of computing the Walsh-Hadamard Transform (WHT) of an $N = 2^n$-dimensional signal whose WHT is K-sparse, when the sparsity parameter $K = O(N^\delta)$ scales sub-linearly in N for some $0 < \delta < 1$. In [2], a sparse graph code approach based on controlled aliasing was proposed to recover the K non-zero WHT coefficients with high probability, using $O(K \log_2(\frac{N}{K}))$ samples and $O(K \log_2(K) \log_2(\frac{N}{K}))$ computations. In this chapter, we propose modifications to the algorithm in [2] and through density evolution analysis and simulations show that the proposed modification substantially improves the scaling constants of the

sample and computational complexities (in some case, it is less than one-third). The main improvement results from the proposed algorithm being able to detect 2 non-zero coefficients at each check node instead of just detecting 1 non-zero coefficient at check node as in [2].

1.4 Pattern Matching

Finally in Chapter 5, we consider the problem of querying a string (or, a database) of length N bits to determine all the locations where a substring (query) of length M appears either exactly or is within a Hamming distance of K from the query. We assume that sketches of the original signal can be computed off line and stored. Using suitable modifications to sparse Fourier transform computation based approach introduced in Chapters 2 and 3, we show that all such matches can be determined with high probability in sub-linear time. Specifically, if the query length $M = O(N^{\mu})$ and the number of matches $L = O(N^{\lambda})$, we show that for $\lambda < 1 - \mu$ all the matching positions can be determined with a probability that approaches 1 as $N \to \infty$ for $K \leq \frac{1}{6}M$. More importantly our scheme has a worst-case computational complexity that is only $O\left(\max\{N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N\}\right)$, which means we can recover all the matching positions in *sub-linear* time for $\lambda < 1 - \mu$. This is a substantial improvement over the best known computational complexity of $O\left(N^{1-0.359\mu}\right)$ for recovering one matching position by Andoni *et al.* [14]. Further, the number of Fourier transform coefficients that need to be computed, stored and accessed, i.e., the sketching complexity of this algorithm is only $O\left(N^{1-\mu}\log N\right)$. Finally, we present several extensions including optimization for longer query lengths ($\mu > 0.5$), algorithmic improvements for correlated data sources, and a secured matching algorithm in an outsourced pattern matching setting.

2. BACKGROUND

This chapter is devoted to reviewing some fundamentals of signal processing and sparse recovery concepts that would be used throughout the dissertation. The chapter begins by introducing some notations, and then briefly reviewing fundamentals of Fourier and Hadamard transforms. This is followed by a brief review of the Fast Fourier Aliasing based Sparse Transform (FFAST) algorithm, a sparse recovery algorithm that was proposed by Pawar and Ramachandran in [3].

2.1 Notations

We use the following notations throughout the dissertation. We use lowercase letters with an arrow to refer to time domain signals, uppercase letters with an arrow to refer to transform-domain signals and boldface uppercase letters to denote matrices. For example, $\vec{\mathrm{x}} = [x[0], x[1], \cdots x[N-1]]^T$ refers to a time-domain signal and $\vec{X} = [X[0], X[1], \cdots X[N-1]]^T$ refers to a transform-domain signal. Let $\mathbb{F}_2$ denote the finite field containing two elements $\{0,1\}$, and $\mathbb{F}_2^n$ denote the n-dimensional vector space over $\mathbb{F}_2$ with addition and scalar multiplication operations performed element-wise in $\mathbb{F}_2$. We use $x[\vec{\mathrm{m}}]$ and $x[m]$ interchangeably to denote the mth element of $\vec{\mathrm{x}}$, where $\vec{\mathrm{m}} = [m[0], m[1], \cdots, m[n-1]]^T \in \mathbb{F}_2^n$ is the n-bit binary representation of $m \in \mathbb{N}$ with $m[0]$ and $m[n-1]$ being the least significant bit (LSB) and most significant bit (MSB), respectively. We define $supp(\vec{\mathrm{x}}) = \{m : x[m] \neq 0, m = 0, 1, \cdots, N-1\}$ as the support of $\vec{\mathrm{x}}$. Finally, we use $[n]$ to denote the set of integers from 1 to n.

2.2 Fourier Transform Fundamentals

In this section, we review some fundamentals of Fourier transform that lays the foundation for chapters 3 and 5.

Definition 1 (Discrete Fourier Transform (DFT)). *Consider a discrete time domain signal* $\vec{x} = [x[0], x[1], \cdots x[N-1]]^T \in \mathcal{C}^N$ *of length* N*. The* N*-point Discrete Fourier Transform (DFT) of the* $\vec{\mathrm{x}}$ *is* $\vec{X} = \mathcal{F}_N\{\vec{\mathrm{x}}\} = [X[0], X[1], \cdots X[N-1]]^T \in \mathcal{C}^N$*, with the* k*-th component* $X[k]$ *defined as*

$$X[k] := \sum_{n=0}^{N-1} x[n]\ \omega_N^{-kn}, \quad k \in \mathbb{Z}_N \tag{2.1}$$

where, $\omega_N = e^{j\frac{2\pi}{N}}$ is the principal N-th root of unity.

The Fourier transform is invertible, i.e., we can recover the time domain signal $\vec{\mathrm{x}} = {\mathcal{F}_N}^{-1}\{\vec{X}\}$ from its Fourier counterpart $\vec{X}$, with the n-th component $x[n]$ given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\ \omega_N^{kn}, \quad n \in \mathbb{Z}_N \tag{2.2}$$

where, ${\mathcal{F}_N}^{-1}$ is the Inverse Discrete Fourier Transform (IDFT) operator.

2.2.1 Properties of DFT

We now discuss two main properties of DFT that are vital for many parts of this dissertation.

Property 2 (Shift property). *Let* $x[n-n_0]$ *represent the signal* $x[n]$ *circularly shifted by* n_0 *samples. The DFT of* $x[n-n_0]$ *is given by*

$$x[n-n_0] \xrightarrow{N-DFT} \omega^{n_0 k} X[k] \tag{2.3}$$

Property 3 (Downsampling/Aliasing property)**.** *Consider an N length signal $\vec{\mathrm{x}}$ with N-point DFT $\vec{X}$. The M-point DFT of downsampled signal $x_s[m] = x[mL], \quad m = \{0, 1, \ldots, N/L = M\}$ denoted by $\vec{X}_s$, with the lth component $X_s[l]$ given by*

$$X_s[l] = M \sum_{p=0}^{L-1} X[l + pM] \tag{2.4}$$

2.3 Hadamard Transform Fundamentals

In this section, we review some fundamentals of Walsh-Hadamard transform that will be useful in chapter 4.

Consider a time domain signal $\vec{\mathrm{x}} = [x[0], x[1], \cdots x[N-1]]^T \in \mathbb{R}^N$ of dimension $N = 2^n$ with samples $x[\vec{\mathrm{m}}]$ indexed by $\vec{\mathrm{m}} \in \mathbb{F}_2^n$, the binary representation of the integer m.

Definition 4 (Walsh-Hadamard Transform (WHT))**.** *The N-point Walsh-Hadamard Transform (WHT) of a signal $\vec{\mathrm{x}} \in \mathbb{R}^N$ denoted by $\vec{X} \in \mathbb{R}^N$, with the $\vec{\mathrm{k}}$-th component $X[\vec{\mathrm{k}}]$ is defined as*

$$X[\vec{\mathrm{k}}] = \frac{1}{\sqrt{N}} \sum_{\vec{\mathrm{m}} \in \mathbb{F}_2^n} (-1)^{\langle \vec{\mathrm{m}}, \vec{\mathrm{k}} \rangle} x[\vec{\mathrm{m}}], \quad \vec{\mathrm{k}} \in \mathbb{F}_2^n$$

where $\langle \vec{\mathrm{m}}, \vec{\mathrm{k}} \rangle = \sum_{i=1}^{n} m[i]k[i]$ denotes the inner product of the binary vectors $\vec{\mathrm{m}}$ and $\vec{\mathrm{k}}$ over $\mathbb{F}_2$.

Definition 5 (Commuting Permutation Matrix)**.** $\mathbf{\Pi_r} \in \mathbb{F}_2^{n \times n}$ *is a commuting permutation matrix if there exists another permutation matrix $\mathbf{\Pi_c}$ such that*

$$\mathbf{\Pi_r H_n} = \mathbf{H_n \Pi_c}$$

where $\mathbf{H_n}$ *denotes the* $N = 2^n$*-point Hadamard matrix.*

Definition 6 (Subsampling matrix)**.** *For every* $b \in \mathbb{N}$ *and* $b < n$*, the subsampling matrix* $\psi_\mathbf{b}$ *is defined by*

$$\psi_b = [\mathbf{0}_{b\times(n-b)}\ \mathbf{I}_b]^T$$

where, $\mathbf{0}$ *and* $\mathbf{I_b}$ *denotes the matrix of all zeros and* $b\times b$ *identity matrices, respectively.*

2.3.1 Properties of WHT

Property 7 (Shift property)**.** *Let* $\vec{\mathrm{x}}_p$ *denote the p-shifted form of* $\vec{\mathrm{x}}$*, i.e.,* $x_p[\vec{\mathrm{m}}] = x[\vec{\mathrm{m}} + \vec{\mathrm{p}}]$*, then the WHT of* $\vec{\mathrm{x}}_p$ *denoted by* $\vec{X}_p$ *is given by*

$$X_p[\vec{\mathrm{k}}] = (-1)^{\langle\vec{\mathrm{p}},\vec{\mathrm{k}}\rangle}\ X[\vec{\mathrm{k}}].$$

Property 8 (Permutation property)**.** *Let* $\mathbf{\Pi_r}$ *and* $\mathbf{\Pi_c}$ *be permutation matrices defined in Def. 5, and* $\vec{\mathrm{y}}$ *denote the permuted form of* $\vec{\mathrm{x}}$*, i.e.,* $\vec{\mathrm{y}} = \mathbf{\Pi_c}\vec{\mathrm{x}}$*. Then, the WHT of* $\vec{\mathrm{y}}$ *is given by* $\vec{Y} = \mathbf{\Pi_r}\vec{X}$*, i.e.,* $\vec{Y}$ *is also a permuted form of* $\vec{X}$*.*

Proof.

$$\begin{aligned}\vec{Y} &= \mathbf{H_n}\vec{\mathrm{y}}\\ &= \mathbf{H_n}(\mathbf{\Pi_c}\vec{\mathrm{x}})\\ &= \mathbf{\Pi_r}(\mathbf{H_n}\vec{\mathrm{x}}) = \mathbf{\Pi_r}\vec{X}.\end{aligned}$$

□

Property 9 (Property 4 in [2], Downsampling/Aliasing)**.** *Let* $\vec{X}$ *be the WHT of a time domain signal* $\vec{\mathrm{x}}$ *of dimension* $N = 2^n$*, and* $\vec{\mathrm{d}} \in \mathbb{F}_2^n$ *be an arbitrary shift. Then, the* $B = 2^b$*-point WHT of the subsampled signal* $\vec{\mathrm{x}}_{\psi_b,d}$ *with m-th component*

$\vec{\mathrm{x}}_{\psi_b,d}[\vec{\mathrm{m}}] = \vec{\mathrm{x}}[\psi_b\vec{\mathrm{m}} + \vec{\mathrm{d}}]$, $\vec{\mathrm{m}} \in \mathbb{F}_2^b$ *and* $\vec{\mathrm{d}} \in \mathbb{F}_2^n$, *denoted as* $\vec{X}_{\psi_b,d}$ *is given by*

$$X_{\psi_b,d}[\vec{\mathrm{k}}] = \sqrt{\frac{B}{N}} \sum_{\vec{\mathrm{i}} \in \mathcal{N}(\psi_b^T)} (-1)^{\langle \vec{\mathrm{d}}, \vec{\mathrm{i}} \rangle} X[\psi_{\mathbf{b}}\vec{\mathrm{k}} + \vec{\mathrm{i}}]$$

where $\mathcal{N}(\psi_b^T)$ *denotes the nullspace of* ψ_b^T.

2.4 Sparse Recovery

The proposed work in this dissertation leverages exciting recent developments on sparse Fourier transform computation [3, 15, 16, 17]. In particular, the proposed schemes in chapters 3 and 5 build on the Fast-Fourier Aliasing-based Sparse Transform (FFAST) algorithm developed by Pawar and Ramchandran in [3] and, hence, we review this material below.

2.4.1 FFAST Algorithm for Sub-Linear Time Sparse Fourier Transform Computation

Let $\vec{x}$ denote an N-length time domain signal and let $\vec{X}$ denote its N-point discrete Fourier transform. Suppose that we wish to compute $\vec{X}$ given $\vec{x}$, but with the extra knowledge that $\vec{X}$ is K-sparse; i.e., only K values of $\vec{X}$ are non-zero. A naive approach is to use all the N time-domain samples, compute the N frequency coefficients, and subsequently identify the K non-zero locations. The computational complexity of this approach is $O(N \log N)$. Given that this approach seemingly overlooks sparsity altogether, the following question arises naturally: When $K \ll N$, can we find the non-zero locations with much fewer samples and with far lower computational complexity? This question is answered in the affirmative. Indeed, one can use classical spectral estimation techniques such as Prony's method to get the non-zero coefficients and their locations, a process that requires $2K$ samples in time and possesses a computational complexity of approximately K^2. In [3],

Pawar and Ramchandran devise an alternate scheme called the FFAST algorithm and show that, if a very small probability of failure can be tolerated, the recovery process requires $4K$ samples yet the computational complexity is only on the order of $O(K \log K)$. That is, the sample complexity and computational complexity can both be sub-linear in N.

The key novelty in the FFAST algorithm is the connection that it makes between spectral estimation and peeling-based decoding of low density parity check (LDPC) codes. The FFAST algorithm cleverly uses a Chinese Reminder Theorem (CRT) guided sub-sampling operation to induce aliasing artifacts in the spectrum that look like the parity check constraints of good error-correcting codes like LDPC codes. This enables the use of computationally-efficient decoding algorithms and permits analysis of the the FFAST framework to leverage tools from the design and analysis of codes on graphs. Specifically, it is shown that when the K non-zero coefficients are chosen uniformly at random among the N coefficients, the ensemble of reduced graphs obtained in the FFAST algorithm case is identical to an LDPC code ensemble and, hence, density evolution can be used to obtain the thresholds for the FFAST algorithm.

Consider the product form where $N = p_1\, p_2 \ldots p_d$. To keep the discussion simple, we restrict our attention to the case of $d = 2$, and we explain the idea with an example with $N = 4 \times 5 = 20$. See the block diagram shown in Figure 2.1. Instead of using all the samples, in the FFAST algorithm, the signal is first sub-sampled by p_1, thereby creating a p_2 point signal $\vec{\mathrm{x}}_s = (x[0], x[p_1], x[2p_1], \ldots, x[(p_2 - 1)p_1)$. Let $\vec{X}_s$ denote the p_2-point DFT of $\vec{\mathrm{x}}$. Next, the signal is delayed $\vec{\mathrm{x}}$ by t_1 (=1 in the example) and then sub-sampled by $p_1 = 4$ to obtain $\vec{\tilde{x}}_s$ and let its DFT be denoted by $\vec{\tilde{X}}_s$. Similarly, $\vec{\mathrm{x}}$ and a shifted version of $\vec{\mathrm{x}}$ are sub-sampled by $p_2 = 5$ to obtain two sequences $\vec{z}_s$ and $\vec{\tilde{z}}_s$. Altogether, the following four sequences and their

corresponding DFT's are generated as shown in Figure 2.1.

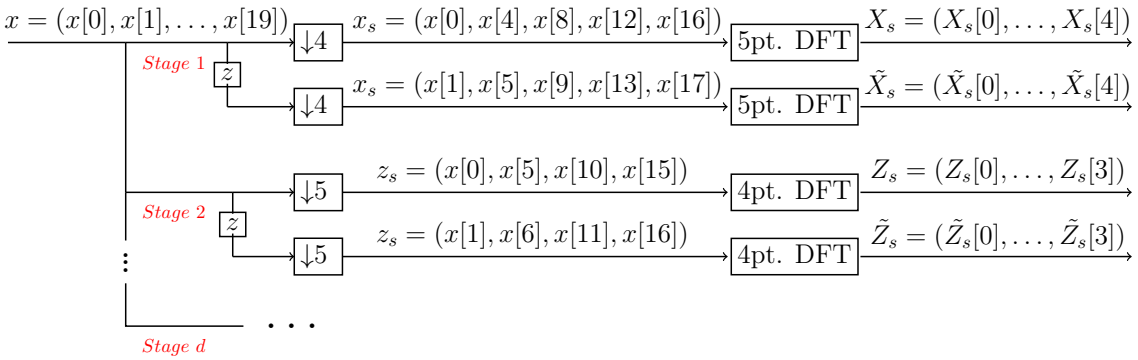

Figure 2.1: Block diagram of a FFAST decoder. Note that this figure was reproduced from [3] © 2013 IEEE.

Sub-Sampling and Aliasing

Notice that, using properties 2 and 3 the relationship between $X[k]$'s and the four aliased DFT coefficients are given by

$$X_s[k] = \sum_{i=0}^{p_1-1} X[k+ip_2], \quad \tilde{X}_s[k] = \sum_{i=0}^{p_1-1} e^{-j\frac{2\pi t_1(k+ip_2)}{N}} X[k+ip_2], \quad 0 \le k \le p_2 - 1$$

$$Z_s[k] = \sum_{i=0}^{p_2-1} X[k+ip_1], \quad \tilde{Z}_s[k] = \sum_{i=0}^{p_2-1} e^{-j\frac{2\pi t_1(k+ip_1)}{N}} X[k+ip_1], \quad 0 \le k \le p_1 - 1$$

For the example of $d = 2$, $N = 5 \times 4 = 20$, the aliasing equations are given below.

$$X_s[0] = X[0] + X[5] + X[10] + X[15] \quad Z_s[0] = X[0] + X[4] + X[8] + X[12] + X[16]$$

$$X_s[1] = X[1] + X[6] + X[11] + X[16] \quad Z_s[1] = X[1] + X[5] + X[9] + X[13] + X[17]$$

$$X_s[2] = X[2] + X[7] + X[12] + X[17] \quad Z_s[2] = X[2] + X[6] + X[10] + X[14] + X[18]$$

$$X_s[3] = X[3] + X[8] + X[13] + X[18] \quad Z_s[3] = X[3] + X[7] + X[11] + X[15] + X[19]$$

$$X_s[4] = X[4] + X[9] + X[14] + X[19]$$

$$x_s[n] = [x[0]\ x[4]\ x[8]\ x[12]\ x[16]] \quad \Leftrightarrow \quad X_s[k] = [X_s[0]\ X_s[1]\ X_s[2]\ X_s[3]\ X_s[4]]$$

$$\tilde{x}_s[n] = [x[1]\ x[5]\ x[9]\ x[13]\ x[17]] \quad \Leftrightarrow \quad \tilde{X}_s[k] = [\tilde{X}_s[0]\ \tilde{X}_s[1]\ \tilde{X}_s[2]\ \tilde{X}_s[3]\ \tilde{X}_s[4]]$$

$$y_s[n] = [x[0]\ x[5]\ x[10]\ x[15]] \quad \Leftrightarrow \quad Y_s[k] = [Y_s[0]\ Y_s[1]\ Y_s[2]\ Y_s[3]]$$

$$\tilde{y}_s[n] = [x[3]\ x[8]\ x[13]\ x[18]] \quad \Leftrightarrow \quad \tilde{Y}_s[k] = [\tilde{Y}_s[0]\ \tilde{Y}_s[1]\ \tilde{Y}_s[2]\ \tilde{Y}_s[3]]$$

The main idea in the FFAST based approach is to exploit the fact that subsampling in the time domain corresponds to aliasing of the spectrum. If the spectrum is sparse, then the aliased spectrum is also likely to be sparse with very few non-zero coefficients aliasing (overlapping) with one another. By first computing multiple versions of aliased spectra with different undersampling factors, we can recover the original spectrum if there is a one-to-one mapping between the aliased spectra and the original spectrum. The FFAST algorithm ensures this through the use of the Chinese remainder theorem.

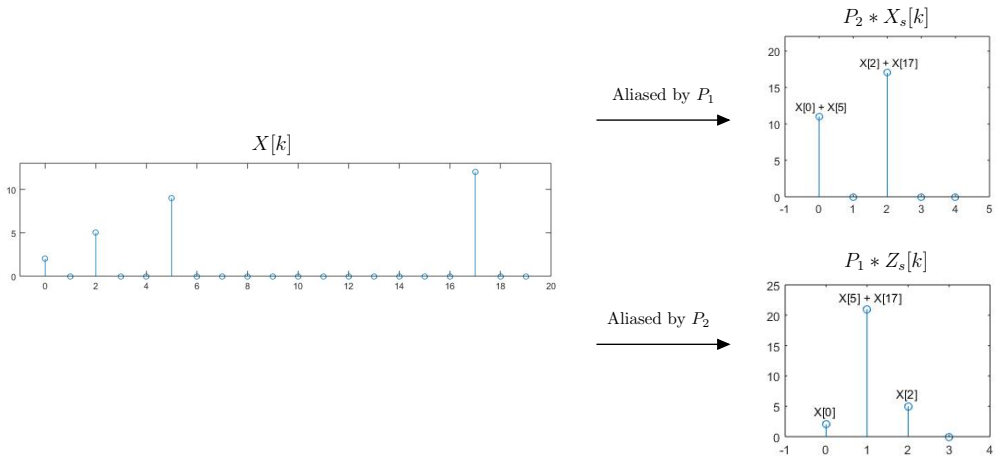

Figure 2.2: An example showing that the spectrum can be recovered from two aliased versions.

Example 1: Consider an example with $N = 20$, where $p_1 = 4, p_2 = 5$ and when the non-zero coefficients are $X[0], X[2], X[5], X[17]$. In Figure 2.2, the non-zero $X[k]$'s and how they appear in the aliased version is shown. This can also be visualized as a bipartite graph as shown in Figure 2.3. The left nodes in the graph represent the non-zero $X[k]$'s and the right nodes represent the (scaled) aliased DFT coefficients $X_s[k]$ and $Z_s[k]$. The degree of the nodes on the right (number of edges connected to it) denotes the number of non-zero DFT coefficients aliasing or adding up.

The idea behind the FFAST algorithm is to successive identify non-zero DFT coefficients by one at a time. Let us first consider $Z_s[2]$ and $\tilde{Z}_s[2]$. Since they refer to aliased spectral coefficients, it can be seen that

$$\begin{aligned} Z_s[2] &= X[2] + X[6] + X[10] + X[14] + X[18] \\ \tilde{Z}_s[2] &= e^{-\frac{j2\pi 2}{N}} X[2] + e^{-\frac{j2\pi 6}{N}} X[6] + e^{-\frac{j2\pi 10}{N}} X[10] + e^{-\frac{j2\pi 14}{N}} X[14] + e^{-\frac{j2\pi 18}{N}} X[18] \end{aligned}$$

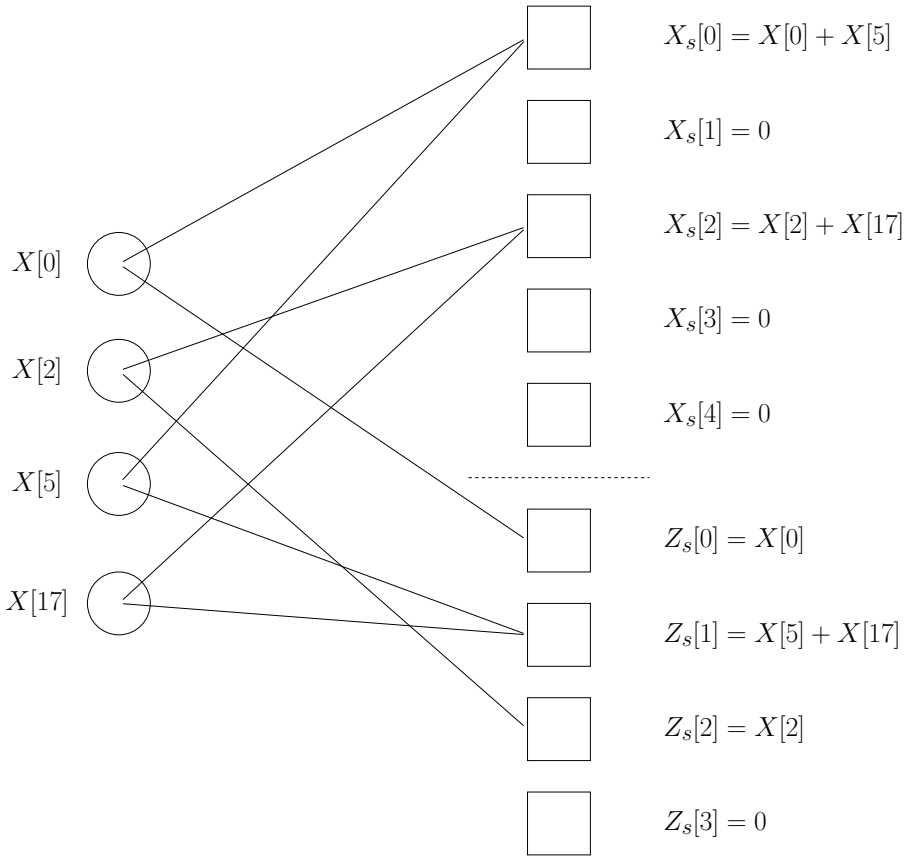

Figure 2.3: An example where the FFAST algorithm is successful.

Let us now consider the pair $(Z_s[2], \tilde{Z}_s[2])$. If $X[2], X[7], X[12]$ and $X[17]$ are all zero, then $(Z_s[2], \tilde{Z}_s[2])$ are both zero and hence we can determine that $X[2], X[7], X[12]$ and $X[17]$ are all zero. If exactly one coefficient among $X[2], X[7], X[12]$ and $X[17]$ is non-zero (referred to as a singleton), then the non-zero coefficient can be recovered from the ratio $\tilde{Z}_s[2]/Z_s[2]$ by extracting the exponent. If more than one of the coefficients is non-zero, then this situation is identifiable, but the coefficients cannot be determined. The algorithm proceeds by determining singletons and substracting those non-zero coefficients from all the right nodes containing the non-zero coefficient. In the example, one can first determine $X[2]$ from $(Z_s[2], \tilde{Z}_s[2])$, then $X[2]$

can be subtracted from $X_s[2]$ and $X[17]$ can be determined from $(X_s[2], \tilde{X}_s[2])$, then $X[5]$ can be determined from $(Z_s[1], \tilde{Z}_s[1])$ and so on. This process can be seen to be identical to peeling the non-zero values similar to erasure decoding of a low density generator matrix (LDGM) code [18] as shown in Figure 2.3.

Example 2: The above approach is greedy, and as can be seen from Figure 2.4, if the non-zero coefficients are $X[2], X[5], X[12]$ and $X[17]$, then there are no singletons and, hence, the greedy procedure gets stuck. Indeed, these are exactly the stopping sets associated with the graphs [18, 3].

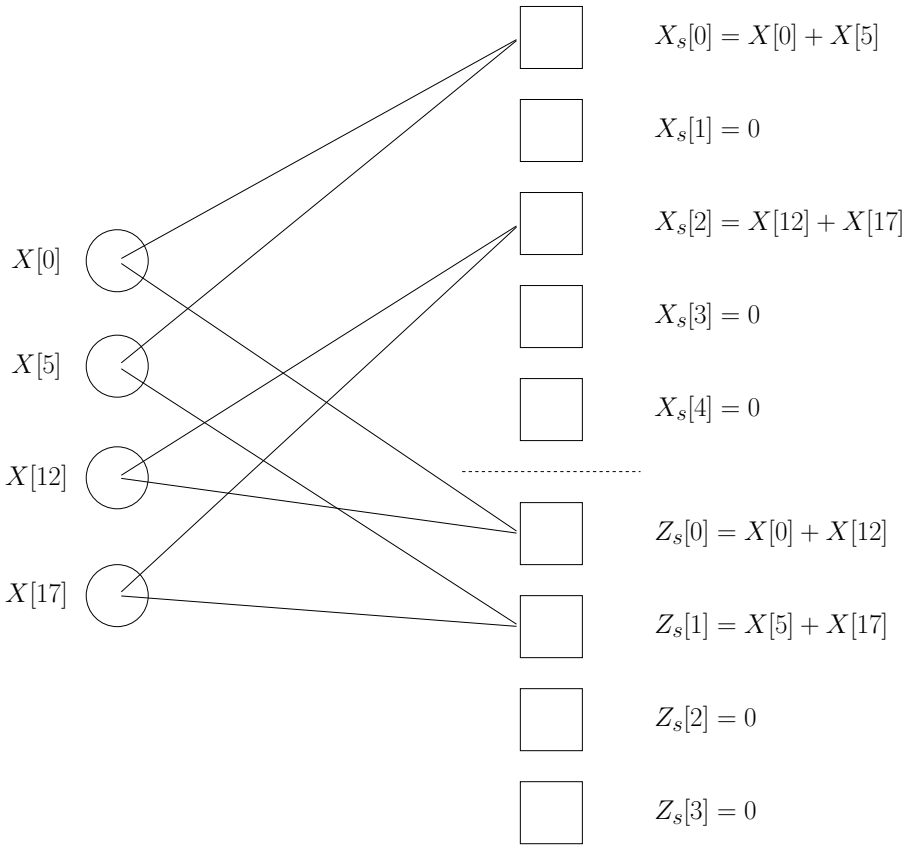

Figure 2.4: An example where the FFAST algorithm is not successful.

By leveraging the connection to decoding of LDPC codes, in [3], Pawar and Ramchandran used tools from the analysis of codes on graphs such as density evolution [18], [19] to show that sharp thresholds exist for computing FFT. That is, in the limit of large N, as long as K scales as N^{α}, $\alpha < 1$, it suffices to take $4K$ samples and with a complexity of $O(K \log K)$. This algorithm succeeds with arbitrarily high probability. Even though we have described only the case when there is no noise in the observations and when the zero coefficients are exactly zero, our interest in Chapter 5 will be in the case when the zero coefficients are small but not exactly zero. The above peeling based decoder has been shown to be robust in such cases as well [20].

3. CONNECTIONS BETWEEN SPARSE FOURIER TRANSFORM COMPUTATION AND DECODING OF PRODUCT CODES [1]

3.1 Introduction and Key Contributions

Let $\vec{x} = (x[0], x[1], \ldots, x[N-1])$ denote a discrete-time signal of length N that is sparse in the Fourier domain with exactly K non-zero Discrete Fourier Transform (DFT) coefficients. Let $\vec{X} = (X[0], X[1], \ldots, X[N-1])$ denote the DFT of $\vec{x}$. Recently, there has been a lot of interest in the design and analysis of computationally-efficient algorithms for computing the location and magnitudes of the K non-zero coefficients from a sub-sampled version of $\vec{x}$ [3], [16], [15].

In this chapter [21], it is shown that in the very sparse ($K = O(N^{\delta}), 0 < \delta \leq 1/3$) regime, the FFAST algorithm with d stages, i.e., with d sub-sampling patterns (refer to [22] for details) is identical to an iterative decoder for a d-dimensional product code with a single error correcting code in each dimension. This connection allows us to leverage some tools that have been developed for the analysis of product codes to rederive the thresholds for the FFAST algorithm. Particularly, Justesen has analysed the thresholds of iterated product codes in [23] for two-dimensional product codes with arbitrary t-error correcting component codes. It is shown that extending this analysis to d-dimensional codes results in exactly the same thresholds as in [3], but it provides an alternate characterization of the threshold. This turns out to involve quantities that typically appear in the results on the emergence of k-cores in random graphs. In the less-sparse ($K = O(N^{\delta}),\ 1/3 < \delta \leq 1$) regime, it is shown that the FFAST algorithm corresponds to an iterative decoder for a carefully defined product

[1] © 2015 IEEE. Reprinted, with permission, from N.T. Janakiraman, S. Emmadi, K. R. Narayanan, K. Ramchandran "Exploring connections between Sparse Fourier Transform computation and decoding of product codes", 53rd Annual Allerton Conference on Communication, Control, and Computing, 2015.

code and we show that such a code can also be analyzed and the thresholds derived based from this analysis are identical to those derived in [3].

Interpreting the FFAST algorithm as decoding of a product code also provides insight into the performance of the FFAST algorithm when non-zero coefficients are not randomly chosen, but are bursty such as what may be encountered in many practical applications like spectrum sensing. This thesis also provides partial results on the recovery of bursty signals when there is exactly one burst or two bursts, i.e., when the spectrum consists of one or two contiguous bands. We provide guaranteed recoverability results for the finite length case and provide thresholds for the 1 and 2 burst cases asymptotically. The performance of the FFAST algorithm for bursty signals appears to be better than those for randomly chosen non-zero coefficients. It is shown that for $d = 2$ stages when there are two bursts, in the limit of $N \to \infty$, perfect recovery is possible when $K < \alpha\sqrt{N}$ for $\alpha \approx 1$. However, under the random sparsity model for $d = 2$, the probability of recovery cannot be made to be arbitrarily close to 1 for any $\alpha > 0$.

3.2 System Model

The main idea in the FFAST framework [3] is to exploit the fact that sub sampling in the time domain corresponds to aliasing of the spectrum. If the spectrum is sparse, then the aliased spectrum is also likely to be sparse with very few non-zero coefficients aliasing (overlapping) with one another. By first computing multiple versions of aliased spectra with different under-sampling factors, we can recover the original spectrum if there is a one-to-one mapping between the aliased spectra and the original spectrum. The FFAST algorithm ensures this through the use of the CRT.

Our system model (shown in Figure 3.1) is a generalized form of the setup pre-

sented in the FFAST framework. In this section, we describe our model and show that FFAST is a special case of this architecture.

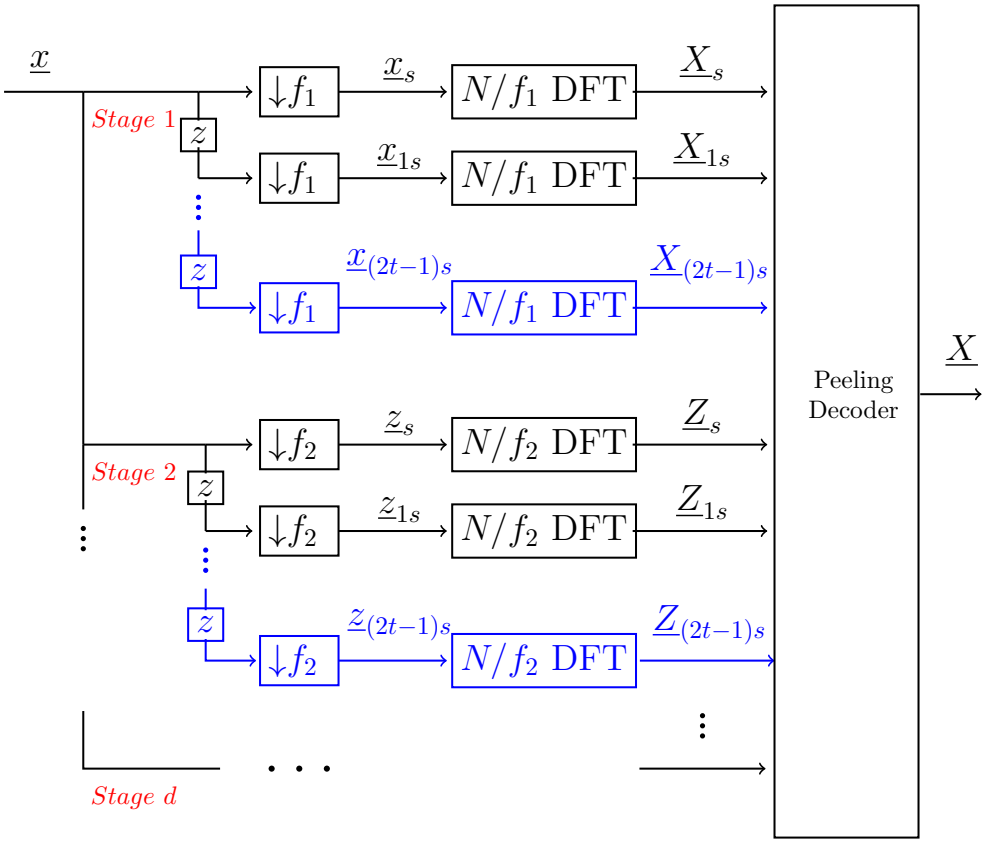

Figure 3.1: Block diagram of an extended FFAST decoder: There are a total of d stages when $N = P_1 P_2 \dots P_d$. $\vec{\mathrm{x}}[n]$ and $2t-1$ shifted versions of $\vec{\mathrm{x}}[n]$ are sub-sampled by f_i in the ith stage and the Fourier transform of the sub-sampled versions (aliased versions of the original spectrum) are computed. A peeling decoder is then used to recover the original DFT coefficients from the aliased spectra. The blue color boxes indicate the extensions to the FFAST algorithm.

The generalized FFAST architecture consists of d stages and each stage with $2t$ branches. This framework computes the N-point DFT of a signal $\vec{\mathrm{x}}$, whose spectrum $\vec{X}$ is purely (noiseless) k-sparse. Let $N = P_1 \times P_2 \times \ldots \times P_d$, where P_is, $1 < i \leq d$, are

pairwise co-prime positive integers. To keep the discussion simple, we will restrict our attention to the case of $d = 2$ and consider only the first two stages in the Figure 3.1 with $N = P_1P_2$, $f_1 = P_1$ and $f_2 = P_2$. In the ith stage, $2t$ sequences are generated by delaying x and downsampling by f_i to form $\vec{\mathrm{x}}_s$ and $\vec{\mathrm{x}}_{js}$ $(j = 1, \ldots (2t-1)s)$, whose N/f_i point DFT yields $2t$ aliased spectrum of $\vec{X}$, denoted by $\vec{X}_s$ and $\vec{X}_{js}$. The relationship between $\vec{X}$ and the $4t$ aliased DFT coefficients are given by

$$\begin{aligned}
X_s[l_1] &= \sum_{i=0}^{P_2-1} X[l_1 + iP_1],\ 0 \le l_1 \le P_2 - 1 \\
X_{1s}[l_1] &= \sum_{i=0}^{P_2-1} e^{-j\frac{2\pi s_1(l_1+iP_1)}{N}} X[l_1 + iP_1],\ 0 \le l_1 \le P_2 - 1 \\
&\vdots \\
X_{(2t-1)s}[l_1] &= \sum_{i=0}^{P_2-1} e^{-j\frac{2\pi s_{2t-1}(l_1+iP_1)}{N}} X[l_1 + iP_1],\ 0 \le l_1 \le P_2 - 1
\end{aligned}$$

$$\begin{aligned}
Z_s[l_2] &= \sum_{i=0}^{P_1-1} X[l_2 + iP_2],\ 0 \le l_2 \le P_1 - 1 \\
Z_{1s}[l_2] &= \sum_{i=0}^{P_1-1} e^{-j\frac{2\pi s_1(l_2+iP_2)}{N}} X[l_2 + iP_2],\ 0 \le l_2 \le P_1 - 1 \\
&\vdots \\
Z_{(2t-1)s}[l_2] &= \sum_{i=0}^{P_1-1} e^{-j\frac{2\pi s_{2t-1}(l_2+iP_2)}{N}} X[l_2 + iP_2],\ 0 \le l_2 \le P_1 - 1
\end{aligned}$$

The above equations can be represented as a Tanner graph with left nodes denoting the DFT coefficients of $\vec{X}$, and the right nodes(checknodes) denoting the $2t$ observations. The Tanner graph for $N = 2 \times 3$ is shown in Figure 3.2. A reduced Tanner graph can be constructed if we retain only the non-zero coefficients in the left

nodes. The removal of zero coefficients from the left nodes removes some edges from the checknodes and hence leaving those checknodes will lesser degree compared to others. A checknode is called a *zeroton* if the degree of the node is zero, *singleton* if the degree is one, and *multi-ton* if the degree is greater than 1. A multiton is called a *t-ton* if the degree is less than or equal to t. Hence all singletons are t-tons for $t \geq 1$.

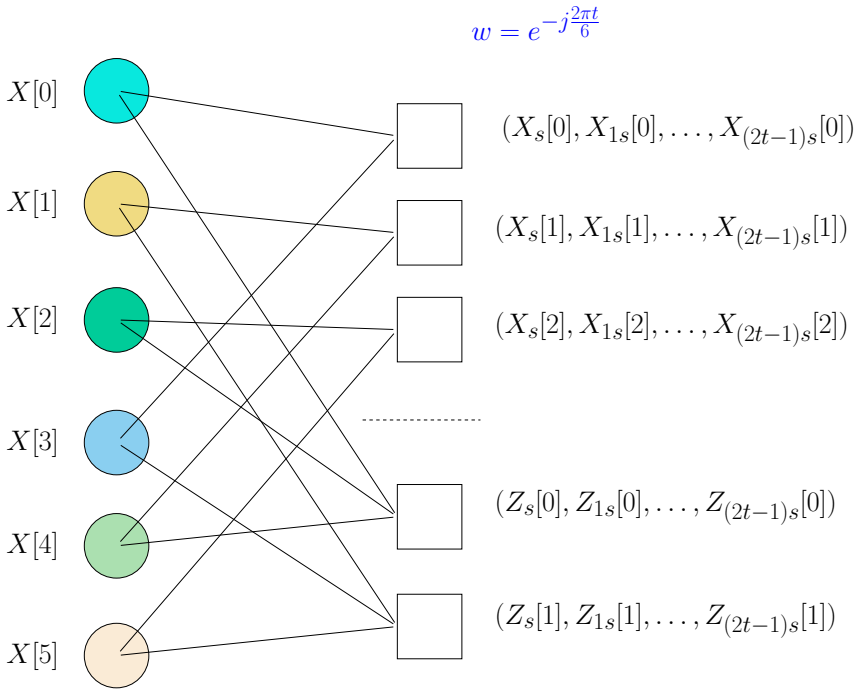

Figure 3.2: Example factor graph for $N = 2 \times 3$. The left nodes represent the coefficients X and the right nodes represent the $2t$ observations.

Consider the lth checknode in the first stage. Let $\beta = e^{-j\frac{2\pi}{N}}$. The checknode observations $(X_s[l], X_{1s}[l], X_{2s}[l], \ldots, X_{(2t-1)s}[l])$ can be rewritten in matrix form as given below.

$$\begin{bmatrix} X_s[l] \\ X_{1s}[l] \\ X_{2s}[l] \\ \vdots \\ X_{(2t-1)s}[l]) \end{bmatrix} = \mathbf{H} \times \begin{bmatrix} X[l] \\ X[l+P_1] \\ X[l+2P_1] \\ \vdots \\ X[l+(P_2-1)P_1]) \end{bmatrix}$$

where $\mathbf{H}$ is given by,

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \beta^{1(l)} & \beta^{1(l+P_1)} & \dots & \beta^{1(l+(P_2-1)P_1)} \\ \beta^{2(l)} & \beta^{2(l+P_1)} & \dots & \beta^{2(l+(P_2-1)P_1)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{(2t-1)(l)} & \beta^{(2t-1)(l+P_1)} & \dots & \beta^{(2t-1)(l+(P_2-1)P_1)} \end{bmatrix}$$

Notice that each parity check represents a t-error correcting Reed-Solomon code in the complex field, that helps the extended FFAST algorithm detect and resolve t-tons. If a multiton checknode has at most t non-zero coefficients participating in it, then decoding the RS code successfully yields the positions and values of the participating coefficients. These coefficients can be peeled off from other checknodes involving it and the next iteration of decoding continues. Since multiple bit-nodes can be identified from a single checknode, the number of iterations required for one complete decoding is less compared to the FFAST algorithm.

If $t = 1$ in the generalized FFAST setup, it gets simplified to the FFAST framework described in Section 2.4.1. Now, each RS code formed is an one-error correcting code that identifies and resolves singletons. The identified variable nodes are peeled off from other participating checknodes.

3.3 Connections between Product Codes and the FFAST Algorithm

3.3.1 Less-Sparse ($K = O(N^\delta), 0 < \delta \leq 1/3$) Regime

To maintain clarity, we will first discuss the connection between product codes and the FFAST algorithm for the less-sparse regime for $d = 2$, and then extend to any $d \geq 3$ and the very-sparse regime. Let us consider the K-sparse spectrum X of length $N = P_1 P_2$, where P_1 and P_2 are co-prime. We will first arrange X in $P_1 \times P_2$ matrix given by X'.

Mapping - There exists a mapping from X' to X, $\mathcal{M} : \{0, 1, \ldots, P_1 - 1\} \times \{0, 1, \ldots, P_2 - 1\} \rightarrow \{0, 1, \ldots, P - 1\}$, which relates $X'(i, j)$ with $X(r + 1)$, where $r = \mathcal{M}(i, j)$ is given by,

$$\begin{aligned} \mathcal{M}(i,j) &\equiv (j-i)bP_2 + i \pmod{N} \\ &\equiv (i-j)aP_1 + j \pmod{N} \end{aligned}$$

where a and b are integers such that $aP_1 + bP_2 = 1$ [24]. The inverse of this mapping from X to X' is given by,

$$(i, j) = \mathcal{M}^{-1}(r) \equiv (r \mod P_2, r \mod P_1).$$

The CRT guarantees that $\mathcal{M}^{-1}$ is indeed a one-to-one and on-to function. The matrix constructed using the above mapping for $N = 4 \times 5$ is shown in Figure 3.3.

From Eqn. 3.1, it can be seen that the l_1th coefficient of $(X_s[l_1], \tilde{X}_s[l_1])$ pair can be used to determine a singleton from the set of values $\{X[l_1 + iP_1],\ 0 \leq l_1 \leq P_2 - 1\}$. The l_1th column of X' corresponds to a single error correcting code. Similarly, the l_2th row of X' corresponds to a single error correcting code that checks the set of values $\{X[l_2 + jP_2],\ 0 \leq l_2 \leq P_1 - 1\}$. The Chinese remainder theorem ensures that

for every r, the $\mathcal{M}^{-1}$ results in a unique pair (i, j). This essentially means that each symbol $X'(i, j)$ participates in exactly one row and one column or, equivalently, the intersection of every row and every column contains only one symbol. The array X' is what is typically referred to as a product code in the coding literature.

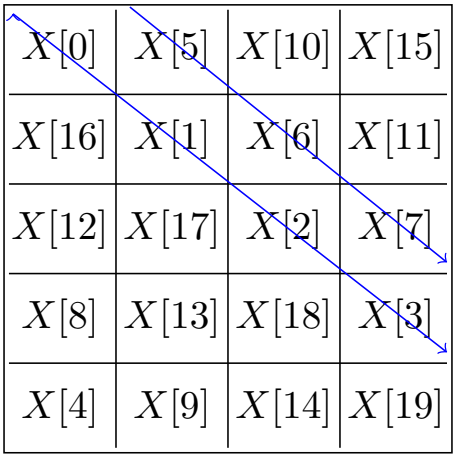

Figure 3.3: Product code matrix for $d = 2$ and $N = 4 \times 5$.

Now that we have established the connection for $d = 2$, let us consider $d \geq 3$ with $N = P_1 P_2 \dots P_d$.

In the less-sparse regime, the down-sampling factors are given by $f_i = P_i,\ 1 \leq i \leq d$. Now, we have d sets of parity check constraints, one from each branch. This could be visualized as a d-dimensional product code given by the following inverse mapping from X to X', namely

$$(i_1, \dots, i_d) = \mathcal{M}^{-1}(r) \equiv (r \mod P_1, \dots, r \mod P_d). \tag{3.1}$$

Every stage of the FFAST creates $N/f_i = \prod_{j \neq i} P_j$ parity constraints, and hence each parity check constraints is described by a 1-dimensional vector of X'. Again the CRT guarantees that the intersection of the d component codes has exactly one symbol. When $d = 3$, the illustration of a representative parity check constraint in the Y-dimension of the product code matrix is shown in Figure 3.4.

3.3.2 Very-Sparse ($K = O(N^\delta), 1/3 < \delta \leq 1$) Regime

We first describe the product code formed for $d = 3$, and then extend to any $d \geq 3$. Consider a K-sparse spectrum, $\vec{X}$, of length $N = P_1 P_2 P_3$, and let us arrange $\vec{X}$ in $P_1 \times P_2 \times P_3$ matrix, X' according to the inverse mapping $\mathcal{M}^{-1}$ in Eqn. 3.1. In the very-sparse regime, the time domain signal, $\vec{\mathrm{x}}$, is down-sampled by down-sampling factors $f_1 = P_2 P_3$, $f_2 = P_1 P_3$ and $f_3 = P_1 P_2$. The ith stage in the FFAST generates $N/f_i = P_i$ parity check constraints, and each check contains f_i coefficients of X. Let us examine the following parity check equations generated at the first stage of FFAST to understand how it could be visualized in the product code matrix X'.

$$X_s[l] = \sum_{i=0}^{P_2 P_3 - 1} X[l + iP_1], \quad 0 \leq l \leq P_1 - 1 \tag{3.2}$$

$$\tilde{X}_s[l] = \sum_{i=0}^{P_2 P_3 - 1} e^{-j\frac{2\pi t_1 (l + iP_1)}{N}} X[l + iP_1], \quad 0 \leq l \leq P_1 - 1 \tag{3.3}$$

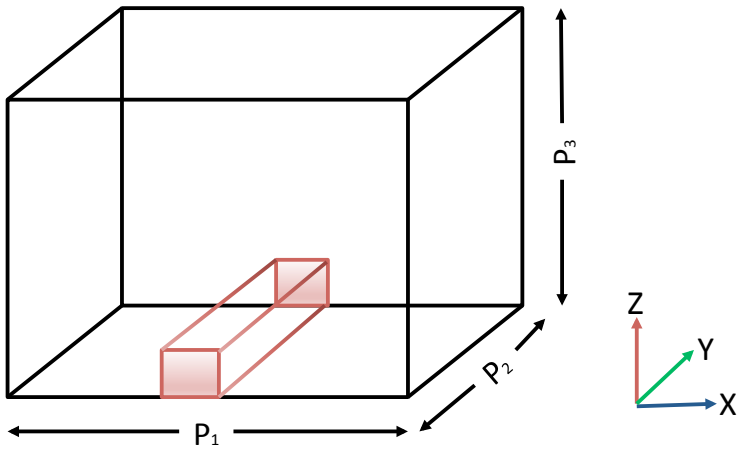

Figure 3.4: Illustration of a representative single-error correcting code in dimension Y, with $d = 3$ and operating in less-sparse regime.

Note that the lth coefficient of the $(X_s[l], \tilde{X}_s[l])$ pair can be used to determine a singleton from the set of values $\{ X[l+iP_1],\ 0 \leq i \leq P_2P_3-1\}$ and hence every check is an one-error correcting code by itself. Notice that elements of $X[l]$ in Eqn. 3.2 lie on a plane in X', where each element on the plane has a remainder l when divided by P_1. Similarly, every check for $d = 3$ in the very-sparse regime can be visualized as a plane in the matrix X' as shown in the Figure 3.5. Every coefficient of $\vec{X}$ participates in three checks (planes), one in each branch(dimension), and the intersection of any two planes formed at different stages gives a parity check generated in less-sparse regime.

Now that we have established the connection for $d = 3$, it can be extended to any $d \geq 3$. Consider a K-sparse signal of length $N = P_1P_2 \dots P_d$ and the down-sampling factors at ith stage of FFAST, $f_i = \prod_{j \neq i} P_j, 1 \leq i \leq d$ and $1 \leq j \leq d$. Let X' be $P_1 \times P_2 \times \dots \times P_d$ matrix whose elements are arranged from X according to $\mathcal{M}^{-1}$. Each parity check equation in this case represents a $(d-1)$-dimensional vector of

X' and every coefficient of X participates in d such vectors, one from each stage of FFAST.

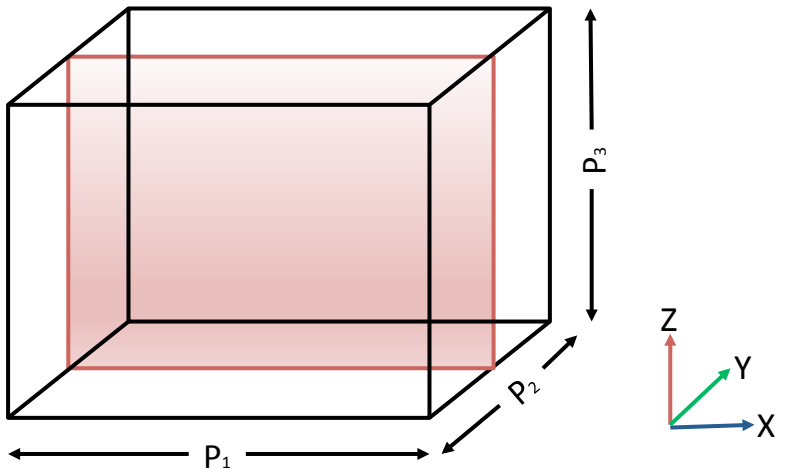

Figure 3.5: Illustration of a representative single-error correcting code in dimension Y, with $d = 3$ and operating in very-sparse regime.

3.3.3 Decoding

It can be seen that the problem of recovering of the K non-zero coefficients is identical to the problem of recovering the errors in a product code assuming the all-zeros codeword is transmitted through a channel that randomly adds K-errors with complex values. It can be seen that an iterative decoding algorithm which decodes all the row codes followed by the column codes in an iterative fashion can be used and this is equivalent to the FFAST algorithm [3]. The iterative process continues until the decoder can not change the codeword(X') anymore from the previous iteration. The decoder stops if all the elements of X' are zero or a non-decodable pattern which contains union of stopping sets (formally defined below) of different sizes is

encountered.

Definition 10 (Stopping Set)**.** *A stopping set of size t, denoted by $\mathcal{S}_t$, is the set of t points (i, j) in X' that form an non-decodable pattern when an iterative decoder is used for decoding the received product codeword.*

Example 11. *Following is a stopping set of size 4. Consider $N = 4 \times 5$ and the set $\mathcal{S}_4 = \{(0,0), (0,1), (1,0), (1,1)\}$. Notice that in Figure 3.3, with the set of points described in $\mathcal{S}_4$, we see that the first two rows and columns have two elements each. But each row code is only an one-error correcting code and hence leaving the decoder to get stuck at this point.*

3.4 Density Evolution and Thresholds

Justesen gave an analysis for iterative decoding of 2-dimensional product codes with t-error correcting RS component codes in [1]. We will use the same approach and extend the analysis to $d-$dimensional product codes in this section. Analytical results for 3-dimensional product codes are compared with that from existing analysis given in [3]. Throughout this section, we will consider a $d-$dimensional product code with individual component code lengths, $P_1, P_2, \ldots, P_d$, all being approximately of same length N^δ. To simplify the notation, we will assume that $P_1 = P_2 = \ldots = P$. This is only to simplify the notation and when $N \to \infty$, this approximation does not change the thresholds. The analysis is given for the $d-$dimensional product code in less-sparse regime and the thresholds for the very-sparse regime can be derived in a similar manner.

3.4.1 Error Pattern as Random Graph

A $P \times P$ 2-dimensional product code can be described using a bipartite graph with P vertices on left representing P rows and P vertices on right representing P columns

and the weighted edges representing the values of the symbols in the corresponding position. As the decoding is independent of codewords and error values, we can work with just the error patterns which translate to random bipartite graphs of $P + P$ vertices where the edges are randomly chosen corresponding to the positions of the random errors.

It is easy to visualize the 2-dimensional product code by its corresponding bipartite graph, but when we extend it to $d-$dimensional ($d \geq 3$) product code, the graph representation is more involved. Each symbol is to be seen as a higher dimensional *hyper* edge connected to d vertices each coming from each of d sets of P^{d-1} vertices. The whole arrangement can be seen as a $d-$partite graph having dP^{d-1} vertices.

3.4.2 Decoding

Decoding of $d-$dimensional product code is performed by an iterative decoder, which decodes all the arrays in each dimension in each iteration. There are d component codes corresponding to each of d dimensions in the product code. Each dimension will have P^{d-1} codeword arrays that belong to the same component code. In each iteration of decoding, all of these same component codeword arrays are decoded, before moving on to decoding another component code along another dimension in the next iteration. An array that belongs to a certain component code along some dimension is decoded if it has t or less number of errors in it. Otherwise, it stays unchanged.

3.4.3 Analysis

In this sub-section, we will be seeing the threshold analysis of the iterative decoder for the $d-$dimensional product codes. The analysis is extended from the analysis given for 2-dimensional product codes in [1]. The following assumptions are taken to simplify the model for analysis

1. The errors are randomly distributed in each array. So, when $P \gg t$, the number of errors distributed along each array follows Poisson distribution.

2. In each iteration decoding along a dimension will result in reduction in the number of errors, which is randomly spread across all the arrays of other dimensions. In the equivalent error graph picture, removal of edges from the light vertices in one partition will randomly remove the edges of all vertices of other partitions with same fraction. This approximation becomes exact when P becomes asymptotically large.

Let the number of errors in an array along a dimension be distributed according to a Poisson distribution with parameter m. Then the average number of errors removed when that array is decoded is $\sum_{j \leq T} j \mathrm{e}^{-m} m^j / j!$. The remaining vertices with degree more than t will not be disturbed by decoding and so, will follow a truncated Poisson distribution. The following lemma is adopted from [1] to $d-$dimensional case.

Lemma 12. *The degrees of the remaining vertices in a partition will follow a truncated poisson distribution after*

1. *the decoding along another dimension (partition) removes some randomly chosen edges (assumption 2), and*

2. *the decoding along the current dimension removes the edges of the newly formed light vertices,*

if the degrees of the vertices in that partition follow a truncated poisson distribution initially.

Proof. This proof follows on the similar lines of the proof given in [1]. □

Definition 13. *Let us define $\pi(m)$ as follows*

$$\pi(m) = \sum_{j \geq T} \mathrm{e}^{-m} m^j / j!. \tag{3.4}$$

If the number of errors is distributed according to a Poisson distribution with parameter m before decoding the codes in a dimension, the distribution of errors after the decoding step follows a truncated Poisson distribution with the same parameter m. The mean number of errors after the decoding step is $\sum_{j \geq T+1} j \mathrm{e}^{-m} m^j / j! = m\pi(m)$.

The following Lemma describes the evolution of mean number of errors in the d dimensional product codes and is an extension of the theorem presented in [1].

Lemma 14. *If the total number of errors in the received codeword is $W = MP^{d-1}$ initially, the number of errors in each array in each dimension follows a Poisson distribution with mean M. After first iteration of decoding in one dimension, the number of errors distributed in each array of all other dimensions follows a Poisson distribution with new mean given by,*

$$m(1) = M\pi(M).$$

After j iterations of decoding, the number of errors in an array, or the degree distribution, follows truncated Poisson distribution with parameter given by,

$$m(j) = \begin{cases} M\pi(M) \prod_{i=1}^{j-1} \pi(m(j-i)) & \text{if } j < d \\ M \prod_{i=1}^{d-1} \pi(m(j-i)) & \text{if } j \geq d \end{cases} \tag{3.5}$$

Proof. This proof follows Justesen's proof in [1] and is extended for the $d-$dimensional

case. The expected number of errors after first decoding is

$$P^{d-1} \sum_{j \geq T+1} j \mathrm{e}^{-M} M^j / j! = P^{d-1} M \pi(M)$$

using the definition 13. These are randomly distributed along all the arrays (P^{d-1} in number) in each dimension other than the current decoded one. So, the new mean of the number of errors in each array is

$$m(1) = \frac{P^{d-1} M \pi(M)}{P^{d-1}} = M \pi(M).$$

After decoding in stage j, the degree distribution of an array in the current decoded dimension is a truncated Poisson distribution with parameter $m(j-1)$. So, the average number of errors remaining per each array is $m(j-1)\pi(m(j-1))$.

When $j < d$, not all d dimensions are decoded by j^{th} stage, so in every remaining dimension other than the currently decoded, the Poisson parameter of the degree distribution of each array is reduced from its initial value of M to $m(j)$. The reduction factor is equal to the product of reduction factors from the stage 1, i.e.

$$\begin{aligned}
\frac{m(j)}{M} &= \prod_{i=1}^{j} (\text{reduction factor in stage } i) \\
&= \prod_{i=1}^{j} \left(\frac{\text{avg. no. of errors in stage } i}{\text{avg. no. of errors in stage } i-1} \right) \\
&= \prod_{i=2}^{j} \left(\frac{m(i-1)\pi(m(i-1))}{m(i-2)\pi(m(i-2))} \right) \left(\frac{M\pi(M)}{M} \right) \\
&= \frac{m(j-1)\pi(m(j-1))}{M}
\end{aligned}$$

Assuming the hypothesis is true for all the stages until $j-1$, the following is true

by induction.

$$\begin{aligned} m(j) &= M\pi(M)\pi(m(1))\pi(m(2))\dots\pi(m(j-1)) \\ &= M\pi(M)\prod_{i=1}^{j-1}\pi(m(j-i)). \end{aligned}$$

When $j \geq d$, in every remaining dimension other than the currently decoded, the Poisson parameter of the degree distribution of an array is reduced from $m(j-d)$ to $m(j)$. The reduction factor is equal to the product of reduction factors from the stage $j-d+2$, i.e.

$$\begin{aligned} \frac{m(j)}{m(j-d)} &= \prod_{i=j-d+2}^{j} (\text{reduction factor in stage } i) \\ &= \prod_{i=j-d+2}^{j} \left(\frac{\text{avg. no. of errors in stage } i}{\text{avg. no. of errors in stage } i-1} \right) \\ &= \prod_{i=j-d+2}^{j} \left(\frac{m(i-1)\pi(m(i-1))}{m(i-2)\pi(m(i-2))} \right) \\ &= \frac{m(j-1)\pi(m(j-1))}{m(j-d)\pi(m(j-d))}. \end{aligned}$$

Assuming that the hypothesis in (3.5) is true for all stages until $j-1$, the following is true by induction.

$$m(j) = \frac{m(j-1)\pi(m(j-1))}{\pi(m(j-d))} \tag{3.6}$$

$$m(j) = M\prod_{i=1}^{d-1}\pi(m(j-i)). \tag{3.7}$$

□

Theorem 15. *In the limit of large P and fixed t, all $W = P^{d-1}M$ errors are decoded*

by the iterative decoder when

$$M < \min_m \{m/\pi^{d-1}(m)\} \triangleq c_{d,t}$$

Proof. To show this, we first note that $m(j)$ is a non-increasing sequence since the decoder never adds more errors. Secondly, it can also be seen that $\pi(m)$ is a monotonically increasing function for $m > 0$. This can be verified by computing the derivative of $\pi(m)$ from (3.4) and noting that $\pi'(m) = \frac{e^{-t}m^t}{t!}$ which is positive for $m > 0$. Thus, $\pi(m(j-i)) \leq \pi(m(j-d)),\ 1 \leq i \leq d-1$. Hence, from (3.7) we can see that

$$\frac{m(j)}{m(j-d)} = \frac{M\prod_{i=1}^{d-1}\pi(m(j-i))}{m(j-d)} \leq M\frac{\pi^{d-1}(m(j-d))}{m(j-d)}. \tag{3.8}$$

When $M < \min_m\{m/\pi^{d-1}(m)\}$, the right hand side of the above equation is strictly less than 1 implying that $m(j) < m(j-d)$ and, hence, $m(j) \to 0$ as $j \to \infty$. □

At the threshold, on the average, there are $c_{d,t}p^{d-1}$ errors and the total number of checks is dP^{d-1}, which means the number of measurements is $2dP^{d-1}$. Thus, the thresholds in terms of the ratio of the number of measurements to the sparsity that can be recovered is given by $\frac{2dtP^{d-1}}{c_{d,t}P^{d-1}} = \frac{2dt}{c_{d,t}}$.

3.4.4 Decoding in Very-Sparse Regime

We have already seen the interpretation of constructing $K-$sparse signal in very-sparse regime as decoding of a $d-$ dimensional product code with codewords being $(d-1)$ dimensional codeword planes. It is different from the conventional d dimensional product codes as the latter have codewords as 1 dimensional codeword arrays in $d-$dimensions. However, the analysis of decoding is similar in both cases as the process of iterative decoding essentially same for both the cases.

Error Graph

The $d-$dimensional product code with $(d-1)-$dimensional codeword planes has P component codes in each of d dimensions, with individual component code length of P^{d-1}. Each symbol participates in d codeword planes. Since the decoding is independent of codewords and error values, we can work with just the error patterns and their error graphs. A corresponding error graph will have hyper edges, chosen according to the random errors in the error pattern, connecting to the vertices, which represent each codeword plane, coming from d sets of P vertices each. This arrangement is a $d-$partite graph with dP vertices.

Evolution of the Poisson Parameter

Decoding of the $d-$dimensional product code with $(d-1)-$dimensional codeword planes is done by iteratively decoding all planes in each dimension in an iteration before moving on to decoding in another dimension in another iteration. This is similar to decoding of the conventional product code with $1-$dimensional codeword arrays, so a similar analysis can be conducted here.

The main idea will be to model the number of errors in each codeword plane as having a Poisson (or, truncated Poisson) before (or, after) decoding each codeword and to track the evolution of the parameter of this Poisson distribution. Similar to the less sparse regime, here we can assume that when one of the codeword planes is corrected, it uniformly affects the errors in the codeword planes in other dimensions. Hence, the asymptotic analysis will be identical to that for the less sparse regime. The following results, similar to those in Theorem 14 and Theorem 15 can be shown.

Lemma 16. *If the total number of errors in the received codeword is $W = MP$ initially, then the degree distribution, or the number of errors distributed in each $(d-1)-$dimensional plane is according to the truncated Poisson distribution of parameter*

given by equation 3.5

Theorem 17. *In the limit of large P and fixed t, all $W = MP$ errors are decoded by the iterative decoder when*

$$M < \min_{m}\{m/\pi^{d-1}(m)\} \triangleq c_{d,t}$$

At the threshold, on the average, there are $c_{d,t}P$ errors and the total number of checks is dP, which means the number of measurements is $2dP$. Thus, the thresholds in terms of the ratio of the number of measurements to the sparsity that can be recovered is given by $\frac{2dtP}{c_{d,t}P} = \frac{2dt}{c_{d,t}}$. When we calculate $c_{d,t}$ and the corresponding thresholds, $\frac{2dt}{c_{d,t}}$, for $3 \leq d \leq 8$ and $t = 1$, we obtain the values given in Table 3.1, and we observe that they are very close to the thresholds given by $2d\eta$ in Table IV in [3].

d	3	4	5	6	7	8
$c_{d,1}$	2.455	3.090	3.509	3.823	4.072	4.280
$\frac{2d}{c_{d,t}}$	2.4440	2.5891	2.8498	3.1389	3.4381	3.7383

Table 3.1: Threshold values.

3.5 Bursty Signals

In this section, we describe a procedure to analyze the performance of FFAST when the input signals are bursty in nature, and also establish some bounds for $d = 2$.

We will first define some notations required to describe the analysis procedure for signals with two bursts. Note that the term *bursty signals* through out this section refers to signals that have non-zero Fourier coefficients occurring in bursts and it should not be confused with the time domain bursts. Let b be the number of bursts.

3.5.1 Single Burst (b=1)

Theorem 18. *For any finite N and $d = 2$, $b = 1$, any burst of length $K \leq P_1 + P_2 - 1$ is guaranteed to be recoverable*

Proof. Without loss of generality, we could assume that the burst starts at (0,0) in X' because of the cyclic property of DFT, and also assume $P_2 < P_1$. The FFAST decoder fails if there exists a sub-matrix of X' which has all the rows and columns filled with more than two coefficients each. In the single burst case, the minimum l for which this occurs is when the burst wraparounds in such a way that the $P_2 \times P_2$ sub-matrix starting at $(0, 0)$ is filled with two elements in each row and column as shown in Figure 3.6. The length of the burst is sum of lengths of the paths$(1, 2, 3, 4)$ shown Figure 3.6. Note that the paths 1 and 2 cover all the P_1 columns once and hence is of length P_1, and the paths 3 and 4 cover all the rows once and hence is of length P_2. So the sum of the path lengths is $P_1 + P_2$. Note that if l is one less than $P_1 + P_2$, there exists a column that has one element in it that is recoverable and the decoder recovers all other coefficients in the subsequent iterations. □

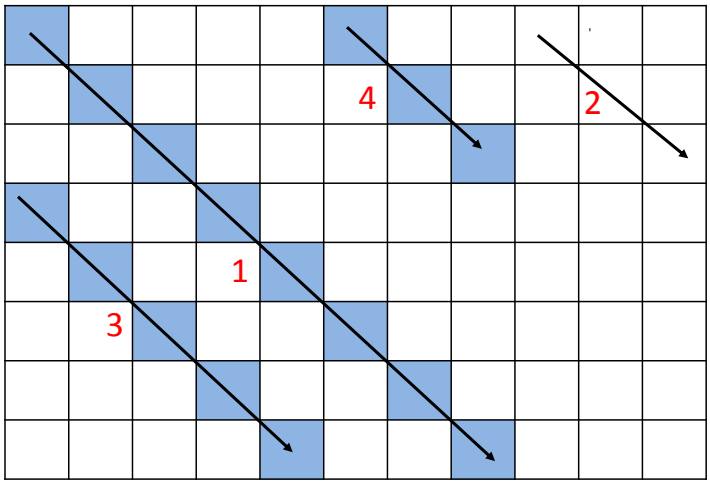

Figure 3.6: Illustration of 1-burst positions in X' for the smallest K for which FFAST fails. The blue colored boxes indicate the positions participating in the stopping set.

3.5.2 Double Burst (b=2)

Let $N = P_1 \times P_2$, and the set of non-zero coefficients that belong to the two bursts be denoted by $\mathcal{B}_1$ and $\mathcal{B}_2$, of lengths k_1 and k_2 respectively. Let the number of symbols (or space) between these two bursts be denoted by s.

We consider the sparse signal $\vec{X}$ in product code structure X', as given by the mapping $\mathcal{M}$ described in Section III. Let us first consider the two bursts $\mathcal{B}_1$ and $\mathcal{B}_2$ such that $\mathcal{B}_1$ starts at the first coefficient of $\vec{X}$, which corresponds to $(0,0)$ in the array X'. A stopping set of size $2n$ contains $2n$ points divided equally in two bursts $\mathcal{B}_1$ and $\mathcal{B}_2$. Consider the case $k_1 < min(P_1, P_2)$, where $\mathcal{B}_1$ has all the points on the diagonal of X' continuously from (0,0) to $(k_1 - 1, k_1 - 1)$. Let us now solve for the points on $\mathcal{B}_2$ that form part of the stopping set of size $2n$.

Definition 19. *The smallest value of k_1 or k_2 such that a stopping set of size $2n$*

occurs in received codeword for some value(s) of space, s, between $\mathcal{B}_1$ and $\mathcal{B}_2$, is defined and denoted as K_n. So, for $k_1, k_2 < K_n$, an $\mathcal{S}_{2n}$ never occurs in the received codeword X with two bursts $\mathcal{B}_1$ and $\mathcal{B}_2$.

Let us now consider a stopping set of size $2n$ denoted as $\mathcal{S}_{2n}$. The points in $\mathcal{B}_1$ would be $(a_1, a_1), (a_2, a_2), \ldots, (a_n, a_n)$. Since $\mathcal{B}_1$ and $\mathcal{B}_2$ form a stopping set, without loss of generality, the corresponding points in $\mathcal{B}_2$ must be n ordered pairs whose first indices are fixed to be $\{a_1, a_2, \ldots, a_n\}$. So, the second indices of points in $\mathcal{B}_2$ will be a permutation, $(b_1, b_2, \ldots, b_n) = \phi_j(a_1, a_2, \ldots, a_n)\ \forall j < n!$ such that $b_i \neq a_i$ $\forall i \in \{1, 2, \ldots, n\}$. So, the points in $\mathcal{B}_2$ are $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$, let us say, occur sequentially, i.e, $\mathcal{M}(a_1, b_1), \mathcal{M}(a_2, b_2), \ldots, \mathcal{M}(a_n, b_n)$ are points in ascending order in mapped $\mathcal{B}_2$ that belongs to X.

Every stopping set of size $2n$ corresponds to a permutation $\phi_j, j < n!$, and the points in $\mathcal{B}_2$ satisfy the following linear equations corresponding to that permutation, as their maps, $\mathcal{M}(a_i, b_i)$ for $i = 1, 2, \ldots, n$, are in increasing order of their position in X.

$$\begin{aligned} b_2 - b_1 \pmod{P_1} &\equiv a_2 - a_1 \pmod{P_2} \\ b_3 - b_2 \pmod{P_1} &\equiv a_3 - a_2 \pmod{P_2} \\ &\vdots \\ b_n - b_{n-1} \pmod{P_1} &\equiv a_n - a_{n-1} \pmod{P_2} \end{aligned} \tag{3.9}$$

The set of equations(Eqn. refeqn:congruence) for a permutation ϕ_j is denoted by $\mathcal{P}(\phi_j)$. Each equation is an equation in congruence, is translated to a linear equation in the following sense. As we are finding the solution for $\{a_1, a_2, \ldots, a_n\} < P_2$, each congruence equation $x \pmod{P_1} \equiv y \pmod{P_2}$ is equivalent to

$$\begin{cases} x = y & \text{if } x, y \geq 0 \\ P_1 + x = y & \text{if } x < 0, y \geq 0 \\ x = y + P_2 & \text{if } x \geq 0, y < 0 \\ P_1 + x = y + P_2 & \text{if } x, y < 0 \end{cases} \tag{3.10}$$

Translation of a congruence equation to a linear equation requires adding of P_1 or P_2 if there is a wraparound. A wraparound occurs when an element of the burst that is not at the end of the burst lies on the boundary of the product code matrix X'. It could be seen from Figure 3.6 that a wraparound occurs whenever a transition is made from the last element of each path to the first element of the next path. In Eqn. 3.10, we will use the first equation if there is no wraparounds, second if there is a column wraparound, and the third equation if a row wraparound occurs.

Theorem 20. *An $\mathcal{S}_{2n}$ occurs first time in X' for $k_1 = k_2 = K_n$ when k_1 and k_2 are increased, individually, from 1 to P_2 (for $P_2 < P_1$), if there exists a solution for $\mathcal{P}(\phi_j)$ in $\{a_1, a_2, \ldots, a_n\} < P_2$ for some permutation ϕ_j.*

Proof. If there exists a solution in $\{a_1, a_2, \ldots, a_n\} < P_2$ for some permutation ϕ_j, then

$$\begin{aligned} K_n &= \min_{\forall \mathcal{P}(\phi_j)} \mathcal{M}(a_n, b_n) - \mathcal{M}(a_1, b_1) \\ &\equiv \min_{\forall \mathcal{P}(\phi_j)} a_n - a_1 \pmod{P_2} \\ &\equiv \min_{\forall \mathcal{P}(\phi_j)} b_n - b_1 \pmod{P_1} \end{aligned}$$

When an $\mathcal{S}_{2n}$ occurs in X', $\mathcal{B}_2$ contains (a_1, b_1) and (a_n, b_n). So we have $k_2 \geq K_n$. Similarly, we have $k_1 \geq K_n$. Hence, an $\mathcal{S}_{2n}$ occurs first time for $k_1 = k_2 = K_n$ when k_1 and k_2 are increased, individually, from 1 to P_2. In that case, the end points of both bursts $\mathcal{B}_1$ and $\mathcal{B}_2$ are in $\mathcal{S}_{2n}$. □

Corollary 21. *An $\mathcal{S}_{2n}$ occurs in X' for $k_1, k_2 \geq K_n$ if there exists a permutation ϕ_j such that $\mathcal{P}(\phi_j)$ have solutions for $(a_1, a_2, \ldots, a_n)$ for some K_n.*

Proof. This is easy to see from the above proof, as when $k_1, k_2 \geq K_n$, $\mathcal{B}_1$ and $\mathcal{B}_2$ can include $\mathcal{S}_{2n}$ for a certain permutation ϕ_j. □

Note: In the above, we observe that the system of equations $\mathcal{P}(\phi_j)$ has n variables, $\{a_1, a_2, \ldots, a_n\}$, in $n-1$ equations. We have one degree of freedom. We chose that to be the starting point of $\mathcal{S}_{2n}$ in $\mathcal{B}_1$. So, we can express all the points that form $\mathcal{S}_{2n}$ with respect to this point. When $k_1 = k_2 = K_n$, this starting point is the starting point of $\mathcal{B}_1$ as in this case the end points of $\mathcal{B}_1$ and $\mathcal{B}_2$ are in $\mathcal{S}_{2n}$ from Theorem 20.

Lemma 22. *For any finite $N = P_1 P_2$ ($P_1 > P_2$), $d = 2$ and $b = 2$, there exist no stopping set of size $2n$ for those s values that create only one wraparound in bursts, if the length of the bursts is $l \leq \frac{(n-1)}{n} P_2$, where n is a factor of P_2.*

Proof. Consider a stopping set of size $2n$ ($\mathcal{S}_{2n}$) formed by the bursts $\mathcal{B}_1$ and $\mathcal{B}_2$ with $\mathcal{B}_1$ starting at $(0, 0)$. Let us assume that only one wraparound happens in $\mathcal{B}_2$. Consider a permutation ϕ_j with $(b_1, b_2, \ldots, b_n) = (a_n, a_1, a_2 \ldots, a_{n-1})$, a cyclic shift of the indices in $\mathcal{B}_1$. According to Theorem 20, if there exists a solution for the equations $\mathcal{P}(\phi_j)$, then there exists a stopping set of size $2n$ in X'. Without loss of generality, set $a_1 = 0$. The set of linear equations corresponding to the chosen ϕ_j is given by

$$
\begin{aligned}
a_2 - 0 &= 0 - a_n + P_2 \\
a_3 - a_2 &= a_2 - 0 \\
a_4 - a_3 &= a_3 - a_2 \\
a_5 - a_4 &= a_4 - a_3 \\
&\vdots \\
a_k - a_{k-1} &= a_{k-1} - a_{k-2} \\
&\vdots \\
a_n - a_{n-1} &= a_{n-1} - a_{n-2}
\end{aligned}
$$

Notice in the first equation, when the congruence is translated to a linear equation form, there is a P_2 term added to the right hand side, and this is to compensate for the one wraparound assumed. On solving the set of linear equations we get $a_k = \frac{k-1}{n} P_2, 2 \leq k \leq n$. Hence the minimum burst length for which there is a stopping set of size $2n$ is $a_n = \frac{n-1}{n} P_2$. Also notice that for a_k's to be valid points in X', they must be integers, and hence, n must be a factor of P_2. □

Example 23. *An illustration for the above lemma for $n = 2$ is shown in Figure 3.7. The colored boxes indicate the position of the stopping set elements.*

Theorem 24. *For any finite $N = P_1 P_2$ ($P_1 > P_2$), $d = 2$ and $b = 2$, all bursts of length $l \leq P_2/2$ are recoverable.*

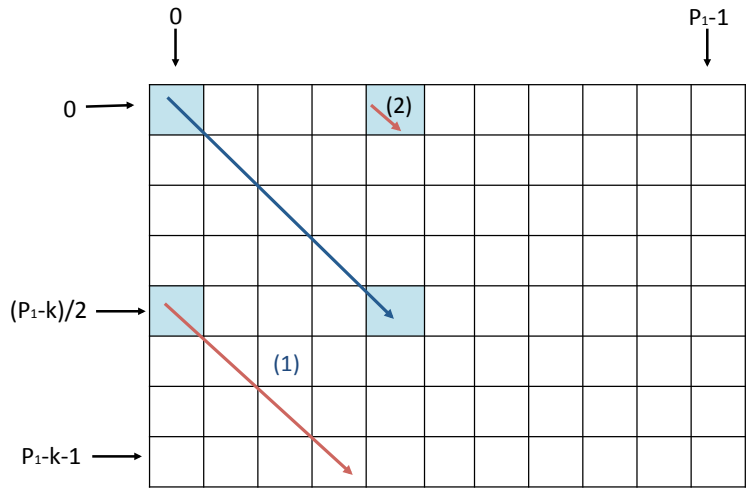

Figure 3.7: Illustration of 2-burst positions in X' for the smallest K, for which FFAST fails, particularly when $N = P_1 \times P_1 - l$, where l is an odd number.

Proof. Considering that only one wraparound is allowed, Lemma 22 states that there is no stopping sets for bursts of length $l \leq \frac{n-1}{n} P_2$. Note that $\frac{n-1}{n} \geq 1/2, \forall n \geq 2$. Hence all separations,$s$, corresponding to one wraparounds can be recovered for $l \leq P_2/2$. Also if $l \leq P_2/2$, all the bursts with two wraparounds can always be recovered, as all the rows and columns other than in the $P_2/2 \times P_2/2$ matrix starting at $(0, 0)$ has at most one coefficient, and also the elements in the $P_2/2 \times P_2/2$ matrix has two bursts that do not wraparound. Hence there are no stopping sets for $l \leq P_2/2$ in both one and two wraparound conditions, leading to the condition where all bursts of length $l \leq P_2/2$ are recoverable. □

This result essentially means that even for $d = 2$, there exists a non-trivial threshold in the bursty case as opposed to the random case.

Even though the guaranteed error correction for two bursts is only $l \leq P_2/2$, the fraction of burst with $P_2/2 \leq l \leq P_2$ that are not decodable appears to be small.

Therefore, we make the following conjecture.

Conjecture: As $N \to \infty$, for $d = 2$, $b = 2$, the fraction of bursts of length $K \leq P_1 + P_2 - 1$ that are not recoverable vanishes as $N \to \infty$.

3.6 Simulation Results

Two kinds of simulations were performed to support the analysis in this chapter.

Bursty signals: Here, we compare the performance of our algorithm over random and bursty signal models. The first set of simulations were conducted with a random choice of the K non-zero coefficients in X and the average probability of error,P_e, curve as a function of the number of non-zero coefficients (sparsity value) K was computed. Here, we define the probability of error of the FFAST algorithm as the fraction of non-recovered non-zero coefficients out of the K non-zero coefficients that were chosen initially. The second set of simulations were conducted for busty signal with two equal bursts of length $K/2$, separated by a separation, s, and the plot for average probability of error, P_e, averaged over all possible separations, s, $0 \leq s \leq N - 2K$, was generated as a function of K. The plots for $N = 250 \times 251$ are shown in the Figure 3.8.

It can be seen from the plot that for the same sparsity level, the performance of the FFAST algorithm is better for the two burst case. It can also be seen that even for the $d = 2$ case, there is a sharp threshold when there are two bursts and the threshold as seen from the simulations matches exactly with that in the conjecture in the previous section. This should be contrasted with the case of randomly chosen non-zero coefficients for which there is no non-trivial threshold for $d = 2$, i.e., there always exists an error floor even when $N \to \infty$.

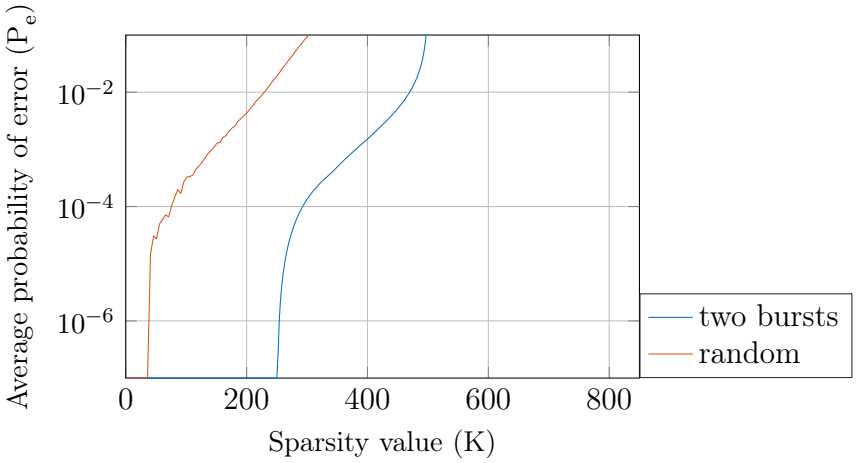

Figure 3.8: Plot of the average probability of error , P_e, as a function of the sparsity value, K, for bursty and random selection of non-zero coefficients ($N = 250 \times 251$ and 1002 samples).

Finite length performance: Another important problem is that the peeling decoder assumes an infinite length for good performance. Here, we analyze the finite length performance of the peeling decoder. The generalized FFAST algorithm (in Section 3.2) proposes key modifications to the FFAST algorithm (in Section 2.1) by replacing the 1-error correcting code in each bin with a t-error ($t >= 1$) correcting Reed Solomon code. Simulation were performed for $N \approx 10,000$ for various values of t and d, and their finite length error performance is reported in Figure 3.9. For $d = 2$, we consider $N = 19 \times 22 \times 23$, $t \in \{1, 2, 3, 4, 5, 6\}$, and for $d = 3$, we consider $N = 100 \times 101$ and $t = 1$. Clearly there is a significant improvement in the performance for $t > 1$.

3.7 Future Work

The work presented here can be extended in many directions and the connection between the FFAST algorithm and product codes can be leveraged for further benefit.

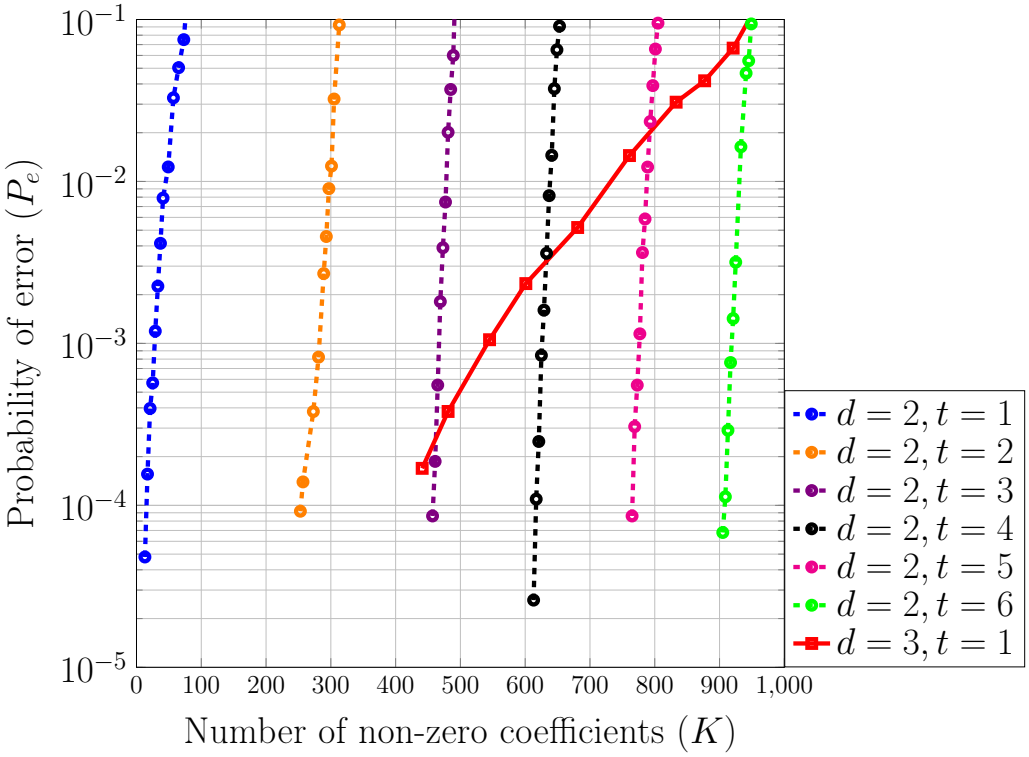

Figure 3.9: Plots of finite length performace of the generalized FFAST algorithm for various d and t values. For $d = 2$, we consider $N = 19 \times 22 \times 23$, $t \in \{1, 2, 3, 4, 5, 6\}$, and for $d = 3$, we consider $N = 100 \times 101$ and $t = 1$.

.

A few possibilities are presented below. It is possible to improve the performance of the FFAST algorithm in the error floor region through the use of post-processing. For example, certain stopping sets can be identified simply by observing the location of the codewords that fail to decode. Based on this, the location of the non-zero coefficients can be identified directly. Such post-processing has been popular in the decoding of product codes. It appears to be possible to analyze the performance of the FFAST algorithm in the presence of miscorrections such as what has been done for product codes in [25]. On the analysis side, it will be useful to determine the rates at which the number of bursts and the length of the bursts can grow with N for probabilistic guarantees in the recovery. Interestingly, this connection also allows

us to leverage the analysis of Pawar and Ramchandran in [3] to compute thresholds for d-dimensional product codes based purely on density evolution. These will be pursued in future work.

4. SPARSE WALSH HADAMARD TRANSFORM

The Walsh-Hadamard Transform (WHT) has been an important tool in signal processing and has been used in the construction of weighing designs, design of spreading sequences for code division multiple access, global positioning systems, compression algorithms and compressed sensing [26]. In this chapter, we consider the problem of computing the WHT of an $N = 2^n$-dimensional signal when the transform is K-sparse. This problem has recently been studied in [2] and [27].

While it is well known that fast algorithms with $O(N \log N)$ computational complexity exist for computing the WHT of an N-dimensional signal, when the transform is known to be sparse, the computational complexity can be substantially reduced. In [2], Scheibler, et al. proposed a sparse graph code based algorithm that can compute the K-sparse WHT of a signal with a sample complexity of $O(K \log(N/K))$ and computational complexity of $O(K \log_2 K \log_2(N/K))$, which are both sub-linear in N. In [27], Cheraghchi and Indyk have designed an algorithm which also has similar complexities as that in [2]. In [28], Li, et al. proposed an extension of the algorithm in [2] to the noisy case.

In this chapter, we consider the noiseless setting as in [2] and we propose modifications to Scheibler, et al.'s algorithm in [2]. The main novelty is in showing that with an appropriate choice of shifts (or, equivalently, appropriate permutations to the columns of the measurement matrix), for the same number of measurements, the sparse graph based algorithm in [2] can be modified to determine 2-sparse signals at each check node in contrast to Scheibler, et. al.'s algorithm which determines only 1-sparse signals at each check node. The algorithm in [27] does not use 2-sparse recovery as well and, hence, differs from our work. We demonstrate through density

evolution analysis and simulations that the proposed modification improves the space and time complexity of the algorithm substantially, sometimes providing as much as a 70% reduction.

4.1 Problem Statement

Consider a time domain signal $\vec{\mathrm{x}} = [x[0], x[1], \cdots x[N-1]]^T \in \mathbb{R}^N$ of dimension $N = 2^n$ with samples $x[\vec{\mathrm{m}}]$ indexed by $\vec{\mathrm{m}} \in \mathbb{F}_2^n$, the binary representation of the integer m. Let $\vec{X} = [X[0], X[1], \cdots X[N-1]]^T \in \mathbb{R}^N$ denote the N-point Walsh Hadamard Transform (WHT, refer to Definition. 4) of $\vec{\mathrm{x}}$.

The noiseless sparse WHT transform computation problem is defined as follows. Given a time domain signal $\vec{\mathrm{x}}$ whose WHT is known to be K-sparse, i.e. the $|supp(\vec{X})| \leq K$ with sparsity $K = O(N^\delta)$ growing sub-linearly in length of the signal for some $0 < \delta < 1$, we are interested in recovering the K positions of the non-zero coefficients in the set $supp(\vec{X})$ and also its values with high probability by using only a subset of samples $x[\mathcal{S}]$ sampled at indices in the set $\mathcal{S} \subseteq \{0, 1, \cdots N-1\}$. The primary goal is to design an algorithm that computes WHT with low sample complexity $|\mathcal{S}|$, and computation time.

4.2 Sparse Graph Based WHT Computation

In this section, we describe the algorithm used to compute the K-sparse Walsh-Hadamard Transform (WHT) $\vec{X}$ of a time domain signal $\vec{\mathrm{x}}$. The algorithm is similar to the algorithm in [2] with the following important differences.

- The choice of shifts is different which results in our sensing matrix (defined later in this section) being a column reordered version of the sensing matrix in [2].

- We propose a new decoding algorithm to determine 2-sparse vectors in each bin

by taking advantage of new shift choices. The algorithm in [2] only identifies 1-sparse vectors in each bin.

Figure 4.1 shows the system architecture which consists of d stages, where the ith stage corresponds to a down-sampling factor $\mathbf{M}_i = \mathbf{\Pi_i}\psi_b$. Each stage has $Q = O(\log N/K)$ branches (shifts), with the jth branch of stage i corresponding to a shift of $\vec{s}_{i,j} \in \{0,1\}^n$. The choice of downsampling matrices and shifts are provided in Sec. 4.3.1 and Sec. 4.3.2 respectively.

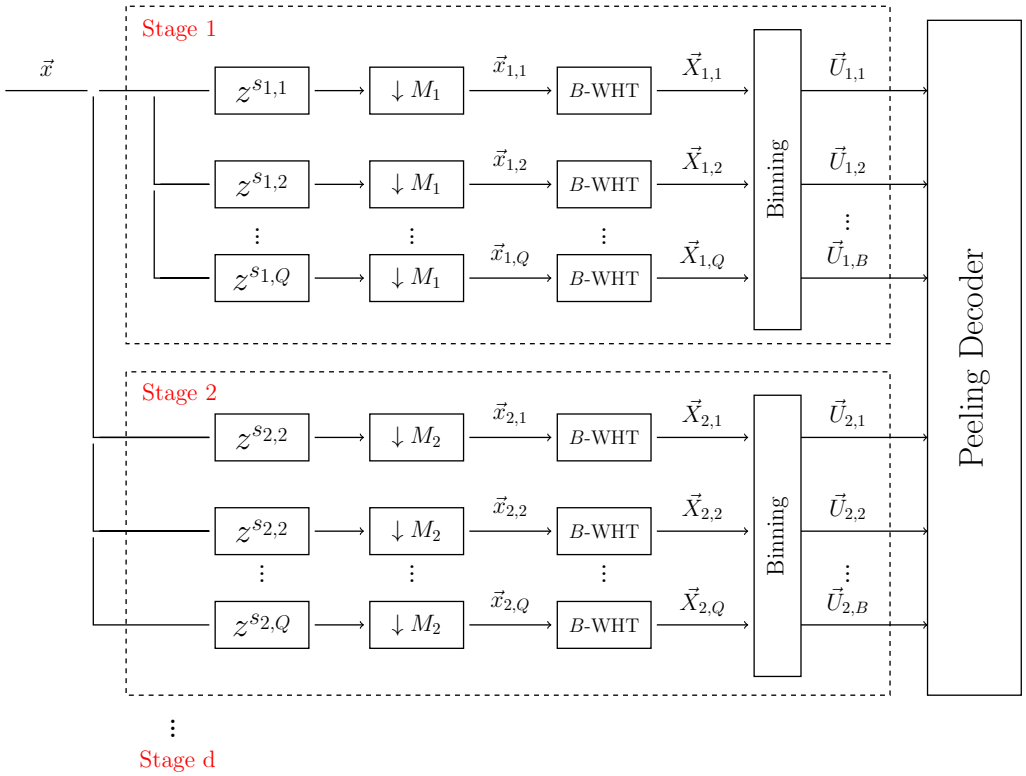

Figure 4.1: Schematic of the sparse WHT algorithm.

Given the input $\vec{\mathrm{x}}$ in branch-j of stage-i, the signal $\vec{\mathrm{x}}$ is shifted by $s_{i,j}$ and then

down-sampled by $\mathbf{M}_i$ to obtain $\vec{\mathrm{x}}_{i,j} = \vec{\mathrm{x}}[\mathcal{I}_{i,j}]$, which is $\vec{\mathrm{x}}$ sub-sampled at B points $\mathcal{I}_{i,j}$ given by

$$\mathcal{I}_{i,j} := \{\vec{s}_{i,j} + \mathbf{M}_i \vec{\mathrm{m}},\ \vec{\mathrm{m}} \in \{0,1\}^b\}.$$

Let $\vec{X}_{i,j}$ denote the B-point WHT of $\vec{\mathrm{x}}_{i,j}$ and let $\vec{U}_{i,k} \in \mathbb{R}^Q$, for $k \in [B]$, be the kth *binned* observation vector of stage-i formed by stacking $\vec{X}_{i,j}[\vec{\mathrm{k}}], j \in [Q]$ together as a vector i.e.,

$$\vec{U}_{i,k} = \left[X_{i,1}[k],\ X_{i,2}[k],\ \cdots,\ X_{i,Q}[k]\right]^T.$$

Note that this gives us a total of B bins (stacked observations) each of length Q in every stage-i. Based on the subsampling/aliasing property of WHT (Property. 9), the observations can be written as

$$\vec{U}_{i,k}[l] = \sqrt{\frac{B}{N}} \sum_{\vec{j} \in \mathbb{F}_2{}^n \ |\ \mathbf{M_i}{}^T \vec{j} = \vec{\mathrm{k}}} (-1)^{\langle \vec{s}_{i,l}, \vec{j} \rangle} X[\vec{j}].$$

This relationship can be expressed in matrix form as

$$\vec{U}_{i,k} = \Phi_{i,k}\ \vec{X}[\mathcal{B}_{i,k}]$$

where $\mathcal{B}_{i,k}$ is the set of coefficients that alias into the (i,k)th bin. We refer to $\Phi_{i,k}$ as a sensing matrix. In general $\Phi_{i,k}$ can change with i and k; however, for our proposed scheme in the next section $\Phi_{i,k}$ will be a constant given in Definition 25.

As shown in [2], the relation between the set of observation vectors $\{\vec{U}_{i,k}, i \in [d], k \in [B]\}$ and $\vec{X}$ can be represented using a Tanner graph, an example of which is shown in Figure 5.3. The nodes on the left, which we refer to as *variable nodes*, represent the N elements of vector $\vec{X}$. Similarly the nodes on the right, which we refer to as *bin nodes*, represent the dB sub-sensing signals (B bin nodes per stage).

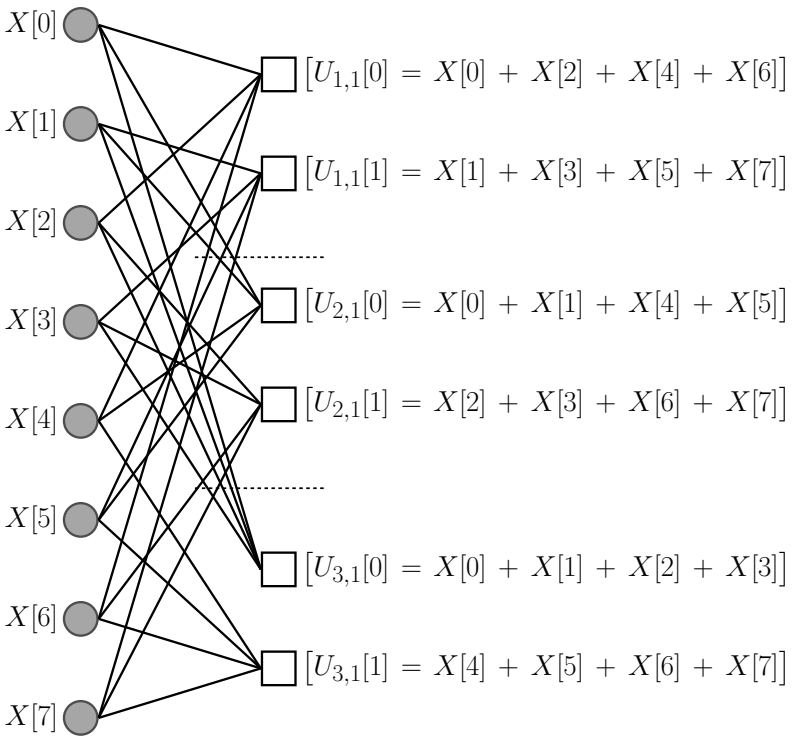

Figure 4.2: Example of a Tanner graph for $N = 2^3$, $d = 3$ and $B = 2$. The variable nodes (gray circles) represent the WHT coefficients $\vec{X}$ and the bin nodes (white squares) represent the binned observation vector $\vec{U}_{i,k}$. The figure illustrates the relationship between $\vec{U}_{i,k}$ and $\vec{X}$.

We will now describe the decoding algorithm which takes the set of observation vectors $\{\vec{U}_{i,k}, i \in [d], k \in [B]\}$, each of length Q, at dB bins as input and estimates the K-sparse $\vec{X}$.

4.2.1 Decoder

Observe from the Tanner graph that the degree of each variable node is d and that of each bin node at each stage is $\frac{N}{B}$. A variable node is referred to as zero if it corresponds to a zero coefficient and non-zero otherwise. We refer to a bin node as *zeroton* (or $\mathcal{H}_z$) if the number of non-zero variable nodes connected to the bin node is zero. The singleton ($\mathcal{H}_s$), doubleton ($\mathcal{H}_d$) and multiton ($\mathcal{H}_m$) bin nodes are defined similarly where the number of non-zero variable nodes connected are one, two, and

greater than two, respectively. The decoder has following three steps in the decoding process.

1. **Bin Classification:** In this step a bin node is classified either as a zeroton or a non-zeroton (singleton or doubleton or multiton). The kth bin node of stage i, denoted by bin node (i, k), with the observation vector $\vec{U}_{i,k}$ is classified as a zeroton ($\mathcal{H}_{i,j} = \mathcal{H}_z$) if all the observations are zeros, i.e., $\vec{U}_{i,k}[l] = 0, l \in [Q]$.

2. **Position Identification:** If a bin node (i, k) is classified as a non-zeroton in the bin classification step, we identify the positions and values of the non-zero variable nodes connected to a singleton or a doubleton using the $2-$sparse recovery algorithm described in Section 4.3.2. This step is different in our approach than in [2] and has one of the two following outcomes:

   - successful decoding of the bin node with one or two non-zero positions and values identified,
   - unsuccessful decoding of the bin node, in which case we classify the bin node as a multiton.

3. **Peeling Process:** The peeling process involves identifying variable nodes connected to the singleton and doubleton bins, decoding its value and removing (*peeling off*) it's contribution from all the bin nodes connected to those variable nodes. The main idea behind this decoding scheme is that, at each iteration, peeling a singleton or a doubleton node off will induce at least one more singleton or doubleton bin and the process of peeling off can be repeated iteratively.

4.3 2-sparse Recovery

Our proposed scheme is based on the insight that the sub-sampling and aliasing scheme introduced in [2] can identify doubletons in addition to singletons (for the same number of measurements) with high probability if the non-zero coefficients are chosen randomly and independenty from a continuous and smooth probability distribution. Further, we propose a computationally simple algorithm that can identify doubletons in addition to singletons.

Definition 25 (2-Sparse Sensing Matrix)**.** *We define a* $(\log_2 N + 1) \times N$ *2-sparse sensing matrix* $\mathbf{\Phi}$ *as a sub-matrix of the Hadamard matrix* $\mathbf{H_n}$*, generated by sampling* $\log_2 N + 1$ *rows from* $\mathbf{H_n}$ *such that the rows are a sequence of* $\{+1, -1\}$ *with different frequencies. Specifically, the frequency of* $(i+1)$*-th row is half of the* i*-th row.*

An example of the sensing matrix $\mathbf{\Phi}$ for $N = 8$ is shown below.

$$\mathbf{\Phi}_{4\times 8} = \begin{bmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & +1 \end{bmatrix}$$

4.3.1 Choice of Sub-sampling Matrices $\mathbf{M_i}$

In less-sparse regime $(1/3 < \delta < 1)$, we choose the number of stages as the smallest $d \in \mathbb{N}$ such that $\delta \leq 1 - 1/d$. Let $l = n/d$ such that $B = 2^b$ with $b = (d-1)l$. The sub-sampling matrix at i-th stage is given as follows.

$$M_i = \begin{bmatrix} \mathbf{I}_{(i-1)l\times(d-i)l} & \mathbf{0}_{(i-1)l\times(i-1)l} \\ \mathbf{0}_{l\times(d-i)l} & \mathbf{0}_{l\times(i-1)l} \\ \mathbf{0}_{(d-1)l\times(d-i)l} & \mathbf{0}_{(d-i)l\times(i-1)l} \end{bmatrix}.$$

The above matrix freezes a set of l-bits in the time domain.

4.3.2 Choice of Shifts

Consider a $N \times N$ Hadamard transform matrix $\mathbf{H_n}$ and let $\mathcal{S} = \{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_Q\}$ be the set of Q chosen shifts. The shifts induce a $Q \times N$ sensing matrix $\mathbf{\Phi}$ formed by selecting a subset of Q rows (one for each shift) from $\mathbf{H_n}$. We choose the shifts such that rows are a sequence of $\{+1, -1\}$ with different frequencies and forms the 2-sparse sensing matrix defined above. Such a choice of shifts always exists in each stage i. Notice that at each stage i and check-node index k, $\mathbf{H_{n-b}}$ can be induced as the sensing matrix if we choose shifts corresponding to rows $r \in \mathcal{B}_{i,1}$, where $\mathcal{B}_{i,k}$ is the index of all aliased coefficients at k-th bin node of stage i. As seen before, we can induce $\mathbf{\Phi}$ from any Hadamard matrix. Hence, there exists a choice of shifts in each stage i namely $\mathcal{S}_i \subset \mathcal{B}_{i,1}$ that selects different frequencies.

With the choice of the shifts and sub-sampling matrices described above, the observations in each bin are a linear combination of a subset of the WHT coefficients. The linear combination is given by the 2-sparse sensing matrix in Defintion 25.

Theorem 26 (2−sparse recovery)**.** *Consider a* $(\log_2 N + 1) \times N$ *sensing matrix* $\mathbf{\Phi}$ *and* $\vec{X}$ *be a* K*-sparse vector whose non-zero coefficients are sampled independently according to a smooth probability distribution. Let* $\vec{U} = \mathbf{\Phi}\vec{X}$ *denote the vector of observations formed by projecting* $\vec{X}$ *onto sensing matrix. Given the observations* $\vec{U}$*, the algorithm 4.3.2 decodes any* $\vec{X}$ *with* $K \leq 2$ *and can declare decoding failure for* $K > 2$*.*

Proof. We prove the theorem by presenting the following algorithm that can recover 1-sparse and 2-sparse vectors while identifying the case that the vector is k-sparse vectors for $k \geq 3$.

2-sparse Recovery Algorithm: Consider the sensing matrix $\mathbf{\Phi}$ and let $X[j_1]$ and $X[j_2]$ be the two non-zero entries in $\vec{X}$ we are interested in recovering. We assume that $X[j_1] \neq \pm X[j_2]$. This assumption is valid as the probability of this event happening is negligible under the assumption that X_j's are chosen independently according to a smooth probability distribution. The observation vector is given by

$$\vec{U} = \mathbf{\Phi}\vec{X}.$$

Notice that the ith measurement is of the form

$$U[i] \;=\; a_i X[j_1] + b_i X[j_2],$$

where a_i,$b_i = \pm 1$. Particularly, for $i = 1$, $a_i = b_i = 1$ and we observe $U[1] = X[j_1] + X[j_2]$. The second measurement $U[2]$ takes one of the forms below.

$$U[2] = \begin{cases} X[j_1] \;+\; X[j_2] & \text{if } a_2 = +1,\ b_2 = +1 \\ -X[j_1] \;-\; X[j_2] & \text{if } a_2 = -1,\ b_2 = -1 \\ X[j_1] \;-\; X[j_2] & \text{if } a_2 = +1,\ b_2 = -1 \end{cases}$$

Notice that $U[2]$ cannot be $-X[j_1] \;+\; X[j_2]$. If $U[2] = X[j_1] \;+\; X[j_2]$ or $U[2] = -(X[j_1] \;+\; X[j_2])$, then we have localized the non-zero values to one half of the indices and since the structure of either half of $\mathbf{\Phi}$ is identical to that of $\mathbf{\Phi}$, we can work inductively. If at some stage k, the non-zero entries are split in the two halves, the above result shows that we are indeed measuring $U[k] = X[j_1] \;-\; X[j_2]$. Therefore, every subsequent measurements can be uniquely identified with an a_i and b_i. Therefore, we can obtain the a_i's and b_i's for every i. Then, we just look for the two columns of $\mathbf{\Phi}$ corresponding to a_i's and b_i's. Given the positions, the values can

be found by solving the system of linear equations induced by any two non-identical observations.

If at any stage of the decoding process we see any measurement other than from $\{\pm U[1], \pm U[k]\}$, we declare a decoding failure and conclude that the number of non-zero coefficients in $\vec{X}$ is greater than two. Also, if $\vec{X}$ is 1-sparse, then the algorithm results in repeated positions $j_1 = j_2 = j$.

□

Example 1. *Consider $N = 4$, $j_1 = 1$ and $j_2 = 2$ and let $X[j_1] = 5$, $X[j_2] = 10$ and $X[0] = X[3] = 0$. Then the observations are given by*

$$\vec{U} = \begin{bmatrix} U[1] \\ U[2] \\ U[3] \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & +1 & -1 \end{bmatrix} \times \begin{bmatrix} 0 \\ X[j_1] \\ X[j_2] \\ 0 \end{bmatrix}$$

As can be seen from the above example, $U[2] \neq \pm U[1]$, meaning there is a split in the second stage ($a_2 = 1,\ b_2 = -1$), i.e., $j_1 \in \{0, 1\}$ and $j_2 \in \{2, 3\}$. In the next stage $U[3] = -U[2]$ and hence $a_3 = -1, b_3 = +1$. Now when we look up for the corresponding columns, we can decode $j_1 = 1$ and $j_2 = 2$. Now that we know the positions, we can solve the system of linear equations corresponding to the first two observations to obtain the values $X[1] = 5$ and $X[2] = 10$.

4.4 Analysis of the Proposed Scheme

We will present a brief analysis for the less-sparse regime, i.e., $\delta > 1/3$. The extension to the very-sparse regime $\delta < 1/3$ can be done based on the results in [21].

Definition 27 (d-dimensional Product Code)**.** *A d-dimensional (n, k) product code*

$\mathcal{P} = \mathcal{C}^1 \otimes \mathcal{C}^2 \otimes \cdots \otimes \mathcal{C}^d$ is the set of $n_1 \times n_2 \times \cdots \times n_d$ arrays of bits such that along the ith dimension, each vector of bits is a codeword from $\mathcal{C}^i$.

Notice that in a d-dimensional product code, each bit is the intersection of d-codewords. For $N = 2^{dP}$, we can arrange the WHT coefficients in a $d-$dimensional array with length 2^P in each dimension. From the description in Section 4.2 and Figure 5.3, it can be seen that every WHT coefficient participates in exactly d bins and each bin can recover 2-sparse vectors. Hence, every coefficient can be thought of as the intersection of d component codes of length $P = N/B$ which are 2-error correcting codes.

4.4.1 Density Evolution

We analyze the performance of the peeling process for $n \to \infty$ and obtain thresholds that characterize the number of measurements needed to successfully recover K non-zero WHT coefficients in the limit of $K, N \to \infty$. The analysis of the peeling process in recovering the K non-zero coefficients with d branches is identical to the analysis of peeling decoder for d-dimensional product codes with 2-error correcting codes in each dimension. Justesen proposed the threshold analysis of iterative decoding for 2-dimensional product codes [23]. The analysis was extended to d-dimensional product codes in [21] and was applied to analyze the measurement complexity of Fast Fourier Aliasing based Sparse Transform(FFAST). We recall the Theorem 15 from Chapter 3.

Lemma 28. *Consider a d-dimensional, length $N = P^d$, product code with t-error correcting component codes, and $K = \mu P^{d-1}$ errors distributed uniformly at random across the d-dimensional array. For the less-sparse and very-sparse regimes, in the limit of $P \to \infty$ and a fixed t, all the K errors can be decoded by the iterative decoder*

when

$$\mu < \min_{m} \frac{m}{\left(\sum_{j \geq t} e^{-m} m^j / j!\right)^{(d-1)}} \triangleq c_{d,t} \tag{4.1}$$

The above result can be used to obtain a threshold on the number of measurements required to compute a K-sparse WHT. Consider a sparse graph based algorithm with d stages and $M_b(d) = dP^{d-1}$ measurements in each branch in each stage. With 2-sparse recovery in each bin, the above result implies that $K = \mu P^{d-1}$ non-zero coefficients can be determined from $(\log_2(N) + 1)dP^{d-1}$ as long as $\mu \leq c_{d,2}$. In order to compare our scheme with the results in [2], we define the $\eta_t(d)$ as the ratio of the number of measurements in each branch to the total recoverable sparsity when each bin can recover a t-sparse signal, i.e.,

$$\eta_t(d) = \frac{M_b(d)}{K} = \frac{d}{c_{d,t}}. \tag{4.2}$$

The results in Schiebler et al., can be recovered by simply setting $t = 1$ in (4.2), whereas our results correspond to $t = 2$. Hence, $\eta_2(d)/\eta_1(d)$ represents the fraction of measurements our proposed scheme requires compared to that in [2].

For a given d and t, we can numerically compute $\eta_t(d)$. Table 4.1 shows the $\eta_1(d)$ and $\eta_2(d)$ for $d = 1, 2, \cdots, 6$. It can be seen from Table 4.1 that our scheme provides substantial savings in the number of measurements required to compute the WHT.

d	2	3	4	5	6
η_1	2.0000	1.2218	1.2949	1.4249	1.5697
η_2	0.5968	0.6440	0.7498	0.8637	0.9788
η_2/η_1	0.2984	0.5271	0.5791	0.6061	0.6236

Table 4.1: Comparison of thresholds for the scheme in [2] and the proposed scheme.

4.5 Simulation Results

We simulated the proposed scheme and compared the performances against the scheme in [2] for the two cases presented below.

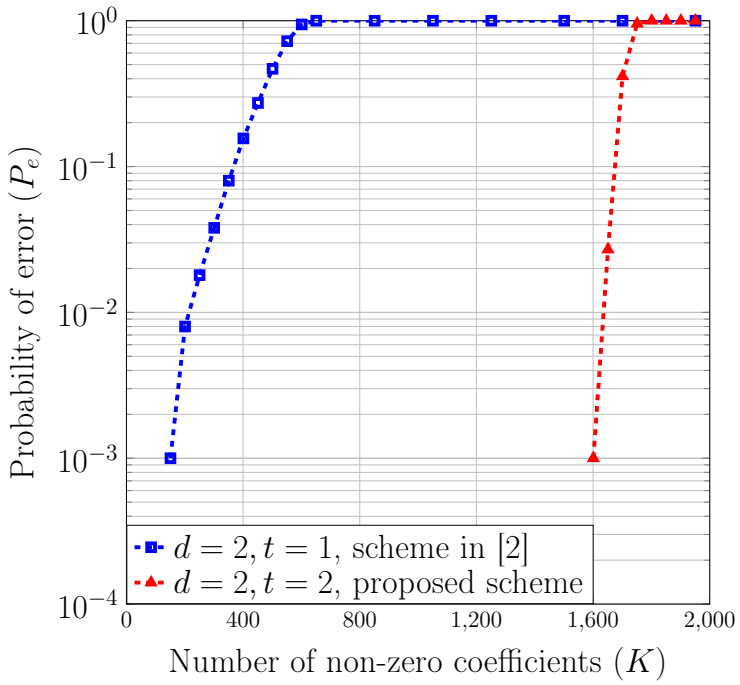

Figure 4.3: Plots of probability of error in recovering the non-zero coefficients vs. Number of non-zero coefficients K. The chosen parameters are $N = 2^{15}$, $d = 2$, $b = 8$ and $M_1 = M_2 \approx 15000$.

- Case 1: We consider the less-sparse regime with $N = 2^{15}$, $d = 2$, $b = 8$. Here we fix the number of measurements as $M_1 = M_2 \approx 15000$, and vary $K \in [200, 2000]$. The plots are shown in Figure 4.3. There are two important observations from the plot. For a fixed probability of error $P_e = 10^{-2}$, our scheme is able to recover $8\times$ more non-zero coefficients compared to the scheme in [2].Our scheme has a sharp threshold for $d = 2$, which is evident around $K = 1650$, whereas the scheme in [2] has a slowly changing error probability.

- Case 2: We consider the less-sparse regime with $N = 2^{15}$, $d = 3$, $b = 10$. Here we fix the number of measurements as $M_1 = M_2 \approx 46000$, and vary $K \in [1500, 6000]$. The plots are shown in Figure 4.4. Again, notice that for a fixed probability of error $P_e = 10^{-2}$, our scheme is able to recover $3\times$ more non-zero coefficients compared to the scheme in [2].

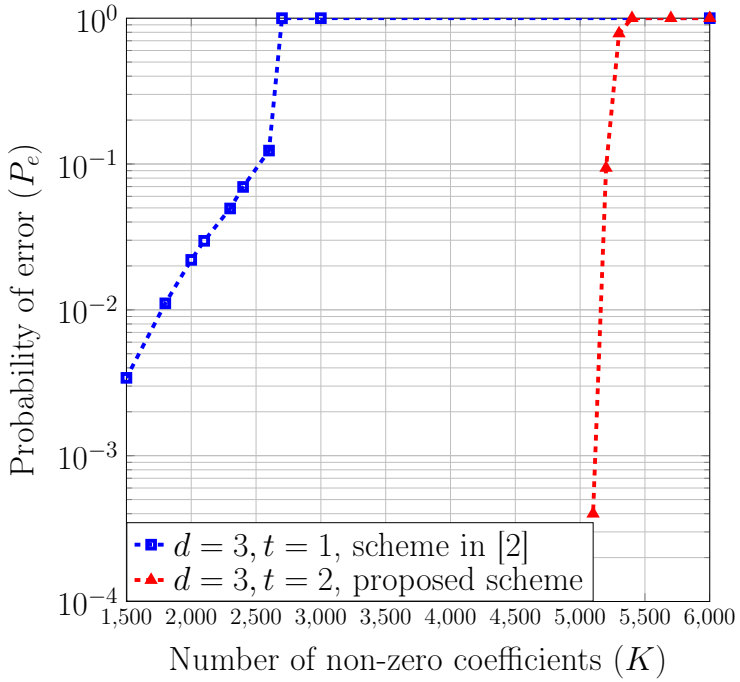

Figure 4.4: Plots of probability of error in recovering the non-zero coefficients vs. Number of non-zero coefficients K. The chosen parameters are $N = 2^{15}, d = 3, b = 10, M_1 = M_2 \approx 46000$.

.

5. PATTERN MATCHING

5.1 Introduction and Problem Statement

We consider the substring/pattern matching problem, which has been studied extensively in theoretical computer science. In this problem, a signal $\vec{\mathrm{x}} := (x[0], x[1], \ldots, x[N-1])$ of length N symbols representing a string, library, or database is available. The objective is to answer queries regarding whether a given string $\vec{\mathrm{y}} := (y[0], y[1], \ldots, y[M-1])$ of length M is a substring of $\vec{\mathrm{x}}$. We are especially interested in the case where a sketch of $\vec{\mathrm{x}}$ can be computed offline and stored, and where the one-time computational complexity of creating the sketch can be amortized over repeated queries or ignored. Moreover, we focus on the random setting in which the $x[i]$'s form a sequence of independent and identically distributed (i.i.d.) random variables, each taking values in $\mathcal{A} \subset \mathbb{R}$. We begin our analysis by restricting our attention to the binary case where $\mathcal{A} := \{\pm 1\}$.[1] Within this context, we consider the following two settings:

Exact Pattern Matching: In the exact pattern matching problem, a substring of length M of $\vec{\mathrm{x}}$, namely $\vec{\mathrm{y}} := \vec{\mathrm{x}}[\tau : \tau + M - 1]$, is obtained by taking M consecutive symbols from $\vec{\mathrm{x}}$ and is presented as a query. This pattern may appear within $\vec{\mathrm{x}}$ in up to L different locations $\tau_1, \tau_2, \ldots, \tau_L$ and the task is to determine all the locations $\tau_i, \forall i \in [1 : L]$. We consider the probabilistic version where our objective is to recover the matching locations with a probability that approaches 1 as $M, N \to \infty$.

Approximate Pattern Matching: In approximate pattern matching, $\vec{\mathrm{y}}$ is a noisy version of a substring, i.e., $\vec{\mathrm{y}} = \vec{\mathrm{x}}[\tau : \tau + L - 1] \odot \vec{\mathrm{b}}$, where $\vec{\mathrm{b}}$ is a noise sequence

[1] Extensions to other alphabets $\mathcal{A} \subset \mathbb{R}$ are straightforward. Extension to the non-i.i.d. case is also possible.

with $b[i] \in \{\pm 1\}$ such that $d_H(\vec{y}, \vec{x}[\tau : \tau + M - 1]) \leq K$. Function d_H denotes the Hamming distance, and $\odot$ represents component-wise multiplication. The objective is to determine all locations τ_i such that $d_H(\vec{y}, \vec{x}[\tau_i : \tau_i + M - 1]) \leq K$ with a probability that approaches 1 as K, M and $N \to \infty$.

We evaluate our proposed algorithm according to two metrics - (i) the space required to store the sketch of $\vec{x}$, which we refer to as sketching complexity, and (ii) the computational complexity in answering the query.

These problems are relevant to many applications including text matching, audio/image matching, DNA matching in genomics, metabolomics, radio astronomy, searching for signatures of events within large databases, etc. The proposed techniques are particularly relevant now due to the interest in applications involving huge volumes of data. Our proposed approach is most useful in the following situations. (i) The string $\vec{x}$ is available before querying and one time computations such as computing a sketch of $\vec{x}$ can be performed offline and stored, and the complexity of computing the sketch of $\vec{x}$ can be ignored. Then, when queries in the form of $\vec{y}$ appear, one would like to decrease the computational complexity in searching for the string $\vec{y}$. (ii) The string $\vec{x}$ is not centrally available, but parts of the string are sensed by different data collecting nodes distributively and communicated to a central server. A search query is presented to the server; and this server must decide whether the string appears in the data sensed by one or more of the distributed nodes and, if present, it must also identify when the queried string appeared. In this latter case, we wish to minimize the amount of data communicated by the nodes to the server and the computational complexity in searching for the string. The proposed formulation is most useful when the query $\vec{y}$ is not a pattern that can be predicted and, therefore, creating a lookup table to quickly identify commonly-occurring patterns will not be effective.

A naive approach to searching for substring $\vec{y}$ in $\vec{x}$ is to first compute the cross-correlation between $\vec{x}$ and $\vec{y}$, which we denoted by $\vec{r} = [r[0], r[1], \ldots, r[N-1]]$ with

$$r[m] = (\vec{x} * \vec{y})[m] \triangleq \sum_{i=1}^{M} x[m+i-1]_M y[i], \tag{5.1}$$

where $[m+i-1]_M = (m+i-1) \mod M$, and subsequently choosing the index m that maximizes $r[m]$. This strategy uses the N samples of $\vec{x}$ and has a super-linear computational complexity of $MN = O(N^{1+\mu})$. A computationally more efficient approach uses the fact that $\vec{r}$ can be computed by taking the inverse Fourier transform of the product of the Fourier transforms of $\vec{x}$ and $\vec{y}^*[-n]$, where $\vec{y}^*[-n]$ is the conjugate of the time reversed version of $\vec{y}$. Such an approach still uses all the N samples of $\vec{x}$, but reduces the computational complexity to $O(N \log N)$. Note that even though the Fourier transform of $\vec{x}$ can be precomputed, the N-point Fourier transform of $\vec{y}$ still needs to be computed online resulting in the $O(N \log N)$ computational complexity.

Both the exact pattern matching problem and the approximate pattern matching problem have been extensively studied in computer science. Three recent articles [14], [29] and [30] provide a brief summary of existing contributions. For the exact matching problem, the Rabin-Karp algorithm solves a more general problem of finding the occurrence of a set of query strings. However, the algorithm has a computational complexity that is at least linear in N. Boyer and Moore presented an algorithm in [31] for finding the first occurrence of the match (only τ_1) that has an expected complexity of $O(N/M \log N) = O(N^{1-\mu} \log N)$, whereas the worst case complexity (depending on τ_1's) can be $O(N \log N)$. For large M, the algorithm indeed has an average complexity that is sub-linear in N. More recently, it has been shown that techniques based on the Burrows-Wheeler transform can be used to solve the exact matching problem with sub-linear time complexity [32] using a storage space of $O(N)$

bits. This problem is well studied under the read alignment setting by the Bioinformatics community [33, 34]. For small $|\mathcal{A}|$, it has the best known complexity; however, the complexity increases with $|\mathcal{A}|$. Further, extensions to approximate matching setting [35] has a complexity that increases exponentially in K and, hence, appear to be infeasible for $K = O(M)$. The Boyer and Moore algorithm has been generalized to the approximate pattern matching problem in [36] with an average case complexity of $O(NK/M \log N)$, which provides a sub-linear time algorithm only when $K \ll M$. In [14], Andoni, Hassanieh, Indyk and Katabi have given the first sub-linear time algorithm with a complexity of $O(N/M^{0.359})$ for $K = O(M)$.

5.2 Main Contributions and Relation to Prior Work

Assume that a sketch of $\vec{\mathrm{x}}$ can be computed and stored. Then for the *exact pattern matching* and *approximate pattern matching* (with $K = \eta M$) problems, with the number of matches L scaling as $O(N^{\lambda})$, we show two algorithms that has

1. Algorithm 1 (refer to Section 5.4):
   - a sketching function for $\vec{\mathrm{y}}$ that computes $O(\frac{N}{M} \log N) \; = \; O\left(N^{1-\mu} \log N\right)$ samples
   - a computational complexity of $O\left(\max\{N^{1-\mu} \log^2 N, N^{\mu+\lambda} \log N\}\right)$
   - a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically in N
2. Algorithm 2 (refer to Section 5.9.1):
   - a sketching function for $\vec{\mathrm{y}}$ that computes S samples given by

$$S \; = \begin{cases} O\left(N^{1-\mu} \log N\right), & \forall \mu \leq 0.5 \\ O\left(\sqrt{N} \log N\right), & \forall \mu > 0.5 \end{cases}$$

.

- a computational complexity of

$$C = \begin{cases} O\left(\max\{N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N\}\right), & \forall \mu \leq 0.5 \\ O\left(N^{\max\{0.5,\ 1-\mu+\lambda\}}\log^2 N\right), & \forall \mu > 0.5 \end{cases}$$

.

- a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically in N

When $L < O\left(\frac{N}{M}\right)$ (i.e. $\lambda < 1 - \mu$), which is typically the interesting case, our algorithm has a *sub-linear time and space complexity.*

There are important differences between this work and [37, 14, 31, 29]. First and foremost, the algorithms used for pattern matching are entirely different. While their algorithms are combinatorial in nature, our algorithm is algebraic and uses signal processing and coding theoretic primitives. Secondly, the system model proposed here differs from the model in [37, 14, 31, 29] in that we allow for preprocessing or creating a sketch of the data $\vec{\mathrm{x}}$. Our algorithm exploits this fact and results in a computational complexity $O(N/M)$ which is better than that in [14] for the approximate pattern matching problem. Finally, we also consider the problem of finding all matches of the pattern $\vec{\mathrm{y}}$ instead of looking for only one match.

This work is inspired by and builds on two recent works by Hassanieh *et al.* in [37] and Pawar and Ramchandran [20]. In [37], Hassanieh *et al.*, considered the correlation function computation problem for a Global Positioning System (GPS) receiver and exploited the fact that the cross-correlation vector $\vec{\mathrm{r}}$ is a very sparse signal in the time domain and, hence, the Fourier transform of $\vec{\mathrm{r}}$ need not be evaluated

at all the N points. In the GPS application, which was the focus of [37], the query $\vec{y}$ corresponds to the received signal from the satellites and, hence, the length of the query was at least N. As a result, the computational complexity is still $O(N \log N)$ (still linear in N) and only the constants were improved in relation to the approach of computing the entire Fourier transform. In a later paper by Andoni *et al.,* [14], a sub-linear time algorithm for shift finding in GPS is presented; however, this algorithm is completely combinatorial and eschews algebraic techniques such as FFT-based techniques.

In [20], Pawar and Ramchandran present an algorithm based on aliasing and the peeling decoder for computing the Fourier transform of a signal with noisy observations for the case when the Fourier transform is known to be sparse. This algorithm has a complexity of $O(N \log N)$ and they do not consider the pattern matching problem. Our algorithm is similar to that of Pawar and Ramchandran's algorithm with one important distinction. We modify their algorithm to exploit the fact that the peak of the correlation function of interest is always positive. This modification allows us to compute the Sparse Inverse Discrete Fourier Transform (SIDFT) with sub-linear time complexity of $O(N^{1-\alpha} \log N)$, $0 < \alpha \leq 1$. One of the main contributions of this work is to show that signal processing and coding theoretic primitives, i.e., Pawar and Ramchandran's algorithm with key modifications can be used to solve the pattern matching algorithm in sub-linear time.

5.3 Notations

We denote signals and vectors using the standard vector notation of arrow over the letter, time domain signals using lowercase letters and the frequency domain signals using uppercase letters. For example $\vec{\mathrm{x}} = (x[0], x[1], \ldots, x[N-1])$ denotes a time domain signal with i^{th} time component denoted by $x[i]$, and $\vec{X} = \mathcal{F}_N\{\vec{\mathrm{x}}\}$ denotes

Symbol	*Meaning*
N	Size of the string or database in symbols
M	Length of the query in symbols
L	Number of matches
μ	Smallest $0 < \mu < 1$ such that $M = O(N^{\mu})$
λ	Smallest $0 < \lambda < 1$ such that $L = ON^{\lambda})$
K	$\max_{\tau} d_H(\vec{\mathrm{x}}[\tau : \tau + M - 1], \vec{\mathrm{y}})$
η	$\frac{K}{M}$
d	Number of stages in the FFAST algorithm
$f_i \approx N^{\alpha}$	Length of smaller point IDFT at each stage-i
$g_i = N/f_i$	Sub-sampling factor in Fourier domain for stage-i
B	Number of shifts (or branches) in each stage
$G = N^{\gamma}$	Number of blocks (for parallel processing)
$\tilde{N} = N^{1-\gamma}$	Length of one block (for parallel processing)

Table 5.1: Parameters and various quantities involved in describing the algorithm the N-point Fourier coefficients of $\vec{\mathrm{x}}$. We denote matrices using boldface upper case letters. We denote the set $\{0, 1, 2 \cdots, N-1\}$ by $[N]$. Table 5.1 introduces the notations we adopt in this chapter.

5.4 System Model

In this section, given the input string $\vec{\mathrm{x}}$ and the query string $\vec{\mathrm{y}}$, we describe our algorithm that finds the matching positions $\mathcal{T} := \{\tau_1, \tau_2, \cdots \tau_L\}$ with sample and time complexities that are sub-linear in N. The main idea exploits the fact that the correlation vector $\vec{\mathrm{r}}$ is sparse (upto some noise terms) with dominant peaks at L matching positions denoted by $\mathcal{T}$ and noise components at $N - L$ positions where the strings do not match.

Consider the correlation signal $\vec{\mathrm{r}}$ in the case of exact matching:

$$r[m] \ = \begin{cases} M, & \text{if } m \in \mathcal{T} \\ n_m, & m \in [N] - \mathcal{T} \end{cases} \tag{5.2}$$

where n_m is the noise component that is induced due to correlation of two i.i.d.

sequence of random variables each taking values from $\mathcal{A} := \{+1, -1\}$. The sparse vector $\vec{\mathrm{r}}$ can be computed indirectly using Fourier transform approach as shown below:

$$\vec{\mathrm{r}} = \underset{\text{III}}{\mathcal{F}_N^{-1}} \{\underset{\text{I}}{\mathcal{F}_N\{\vec{\mathrm{x}}\}} \odot \underset{\text{II}}{\mathcal{F}_N\{\vec{\mathrm{y}}'\}}\} \tag{5.3}$$

where $\mathcal{F}_N\{\cdot\}$ and $\mathcal{F}_N^{-1}\{\cdot\}$ refer to N-point discrete Fourier transform and its inverse respectively, $\odot$ is the point-wise multiplication operation and $y'[n] = y^*[-n]$. Figure 5.1 presents a notional schematic of our Algorithm. As evident from Eq. (5.3), our algorithm for computing $\vec{\mathrm{r}}$ consists of three stages:

- Computing the sketch $\vec{X} = \mathcal{F}_N\{\vec{\mathrm{x}}\}$ of $\vec{\mathrm{x}}$
- Computing the sketch $\vec{Y}' = \mathcal{F}_N\{\vec{\mathrm{y}}'\}$ of $\vec{\mathrm{y}}$
- Computing the IDFT of $\vec{R} = \vec{X} \odot \vec{Y}'$ given $\vec{X}$ and $\vec{Y}'$

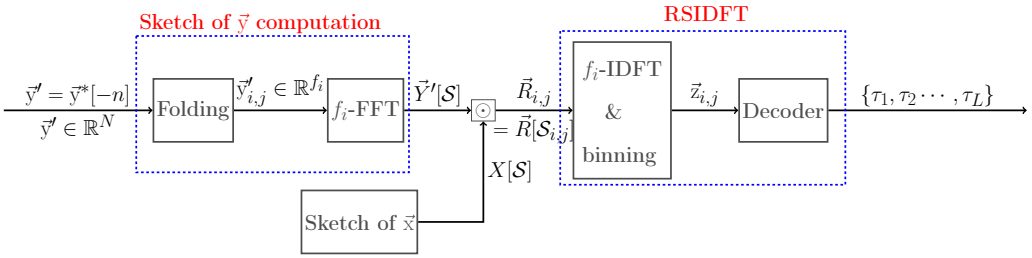

Figure 5.1: Schematic of the proposed scheme using sparse Fourier transform computation.

5.4.1 Sparse Inverse Discrete Fourier Transform

In this section we present Robust Sparse Inverse Discrete Fourier Transform (RSIDFT) scheme that exploits sparsity in the cross-correlation signal $\vec{\mathrm{r}}$ and efficiently recovers its L dominant coefficients. The architecture of RSIDFT is similar

to that of the FFAST scheme proposed in [20], but the decoding algorithm has some modifications to handle the noise model induced in this problem. We will see in Section 5.4.1 how the sketches $\vec{X}$ and $\vec{Y}'$ are computed efficiently but for this section we will focus only on the recovery of the sparse coefficients in $\vec{\mathrm{r}}$ given $\vec{X}$ and $\vec{Y}'$.

Consider the RSIDFT framework shown in Figure 5.2. Let $\vec{R} = \vec{X} \odot \vec{Y}'$ be the DFT of the cross-correlation signal of $\vec{\mathrm{x}}$ and $\vec{\mathrm{y}}$. We begin by factoring N into d relatively prime factors $\{f_1, f_2, \ldots, f_d\}$, where d is a parameter in the algorithm. The design scheme for choosing f_i's for various values of μ such that f_i divides N and $f_i = N^{\alpha} + O(1) \forall\ i \in [d]$ are given in Section 5.4.1. The RSIDFT algorithm consists of *d-stages* with each stage corresponding to a sub-sampling factor of $\frac{N}{f_i}$. In each stage, there are $B = O(\log N)$ *branches* with shifts from the set $\{s_1, s_2, \cdots s_B\}$, where $s_1 = 0$ in the first branch and the rest are chosen uniformly at random from $[N]$.

Given the input $\vec{R}$, in branch j of i^{th} stage of RSDIFT, referred to as *branch* (i, j) for simplicity, RSIDFT sub-samples the signal $\vec{R}$ at

$$\mathcal{S}_{i,j} := \{s_j,\ s_j + g_i,\ s_j + 2g_i,\ \cdots s_j + (f_i - 1)g_i\}, \quad i \in [d], j \in [B] \tag{5.4}$$

where $g_i := \frac{N}{f_i}$ to obtain $\vec{R}_{i,j} := \vec{R}[\mathcal{S}_{i,j}]$. The sub-sampling operation is followed by a f_i-point IDFT in *branch (i,j)* of stage-i to obtain $\vec{\mathrm{r}}_{i,j}$. Notice that $\vec{\mathrm{r}}_{i,j}$ is an aliased version of $\vec{\mathrm{r}}$ due to the property that sub-sampling in Fourier domain is equivalent to aliasing in time domain.

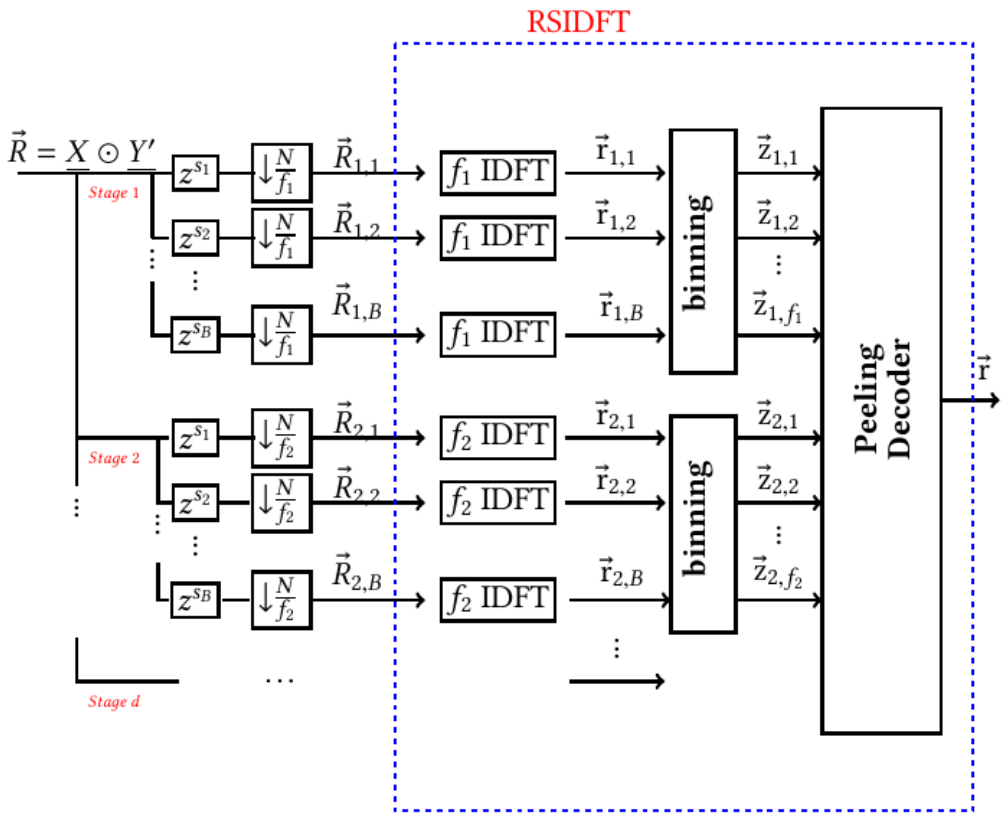

Figure 5.2: RSIDFT Framework to compute inverse Fourier Transform of a signal $\vec{R}$ that is sparse in time domain.

Let $\vec{z}_{i,k} \in \mathbb{R}^B$, for $k \in [f_i] + 1$, be the kth *binned* observation vector of stage-i formed by stacking $\vec{r}_{i,j}[k], j \in [B]$, together as a vector i.e.

$$\vec{z}_{i,k} = \Big[r_{i,1}[k],\ r_{i,2}[k],\ \cdots,\ r_{i,B}[k] \Big]^T$$

Note that this gives us a total of f_i binned observation vectors in each stage-i. Using the properties of Fourier transform, we can write the relationship between the

observation vectors $\vec{z}_{i,k}$ at bin (i,k) and sparse vector to be estimated $\vec{r}$ as:

$$\vec{z}_{i,k} = \mathbf{W}_{i,k} \times \Big[r[k],\ r[k+f_i],\ \cdots,\ r[k+(g_i-1)f_i] \Big]^T \tag{5.5}$$

where we refer to $\mathbf{W}_{i,k}$ as the sensing matrix at bin (i,k) and is defined as

$$\mathbf{W}_{i,k} = \left[\vec{w}^k, \vec{w}^{k+f_i}, \ldots, \vec{w}^{k+(g_i-1)f_i} \right] \tag{5.6}$$

and $\vec{w}^k = \left[e^{\frac{j2\pi k s_1}{N}},\ e^{\frac{j2\pi k s_2}{N}},\ \cdots,\ e^{\frac{j2\pi k s_B}{N}} \right]^T$.

We represent the relation between the set of observation vectors $\{\vec{z}_{i,k}, i \in [1:d], k \in [f_i]\}$ and $\vec{r}$ using a Tanner graph, an example of which is shown in Figure 5.3. The nodes on the left, which we refer to as *variable nodes*, represent the N elements of vector $\vec{r}$. Similarly the nodes on the right, which we refer to as *bin nodes*, represent the $\sum_{i \leq d} f_i$ sub-sensing signals. We will now describe the decoding algorithm which takes the set of observation vectors $\{\vec{z}_{i,k}, i \in [1:d], k \in [f_i]\}$, each of length B, at df_i bins as input and estimates the L-sparse $\vec{r}$.

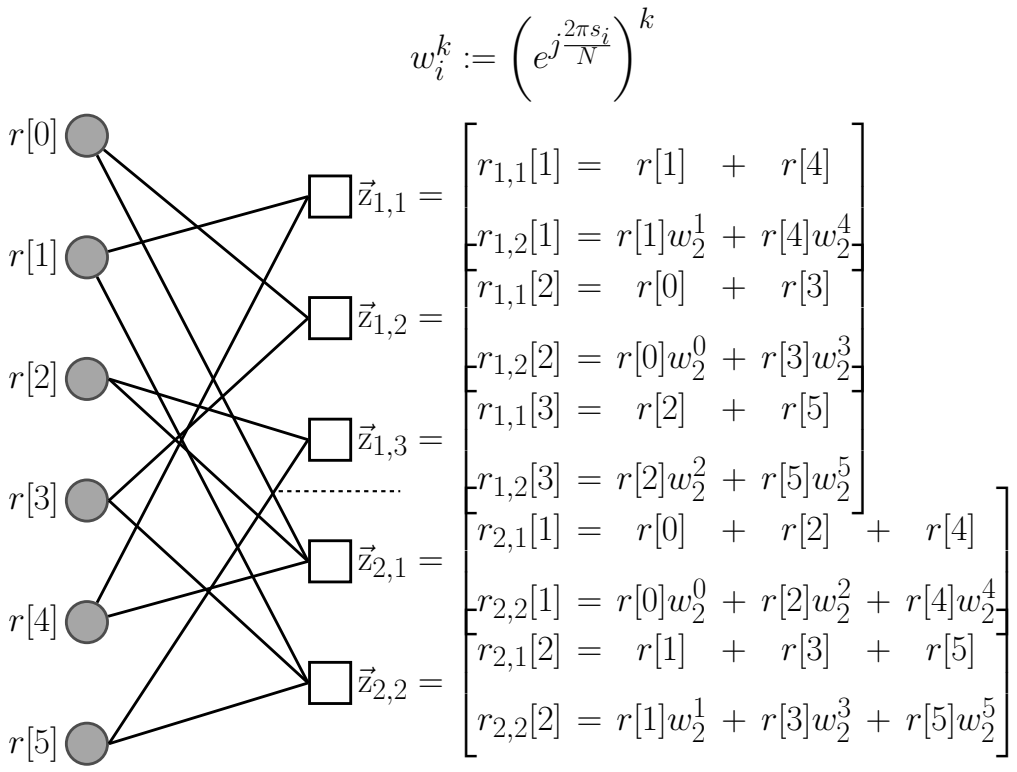

Figure 5.3: Example of a Tanner graph formed in a RSIDFT framework with system parameters being $N = 6$, $f_1 = 2, f_2 = 3$ (i.e., $d = 2$) and $B = 2$. The variable nodes (gray circles) represent the cross-correlation vector $\vec{\mathrm{r}}$ and the bin nodes (white squares) represent the binned observation vector $\vec{\mathrm{z}}_{i,k}$. The figure illustrates the relationship between $\vec{\mathrm{z}}_{i,k}$ and $\vec{\mathrm{r}}$.

Decoder

Observe from the Tanner graph that the degree of each variable node is d and that of each bin node at stage i is g_i. A variable node is referred to as non-zero if it corresponds to a matching position and as zero if it corresponds to a non-matching position. Note that even though the cross-correlation vector value corresponding to a non-matching position is not exactly zero but some negligible noise value we refer to them as zero variable nodes for simplicity. We refer to a bin node as *zero-ton* (or

$\mathcal{H}_z$) if the number of non-zero variable nodes connected to the bin node is zero. The *singleton ($\mathcal{H}_s$), double-ton ($\mathcal{H}_d$) and multi-ton ($\mathcal{H}_m$)* bin nodes are defined similarly where the number of non-zero variable nodes connected are one, two and greater than two, respectively. The peeling decoder has the following three steps in the decoding process.

Bin Classification: In this step a bin node is classified either as a zero-ton or a singleton or a multi-ton. At bin (i, j) the classification is done based on comparing the first observation $z_{i,j}[1]$, which corresponds to zero shift, with a predefined threshold. For $z_{i,j}[1] = z$, the classification hypothesis at bin (i, j), $\widehat{\mathcal{H}}_{i,j}$, can be written as follows:

$$\widehat{\mathcal{H}}_{i,j} = \begin{cases} \mathcal{H}_z & z/M < \gamma_1 \\ \mathcal{H}_s & \gamma_1 < z/M < \gamma_2 \\ \mathcal{H}_d & \gamma_2 < z/M < \gamma_3 \\ \mathcal{H}_m & z/M > \gamma_3 \end{cases} \tag{5.7}$$

where $(\gamma_1, \gamma_2, \gamma_3) = (\frac{1-2\eta}{2}, \frac{3-4\eta}{2}, \frac{5-6\eta}{2})$. Note that for the case of exact matching $\eta = K/M = 0$.

Singleton Decoding: If a bin node (i, j) is classified as a singleton in the bin classification step, we need to identify the position of the non-zero variable node connected to it. This is done by correlating the observation vector $\vec{\mathrm{z}}_{i,j}$ with each column of the sensing matrix $\mathbf{W}_{i,j} := [\vec{\mathrm{w}}^j, \vec{\mathrm{w}}^{j+f_i}, \ldots, \vec{\mathrm{w}}^{j+(g_i-1)f_i}]$ and choosing the index that maximizes the correlation value.

$$\hat{k} = \underset{k \in \{j+lf_i\}}{\arg\max} \ \vec{\mathrm{z}}_{i,j}^{\dagger} \vec{\mathrm{w}}^k$$

where † denotes the conjugate transpose. The value of the variable node connected to the singleton bin is decoded as:

$$\hat{r}[\hat{k}] = M(1-\eta).$$

Note that for the case of exact matching we know the value to be exactly equal to M. But in the case of approximate matching, the actual value of $r[k] \in \{M(1-2\eta), \ldots, M-1, M\}$ and our estimate $\hat{r}[\hat{k}] = M(1-\eta)$ is only approximate. But this suffices for recovering the positions of matches i.e., the indices of the sparse coefficients in $\vec{\mathrm{r}}$.

Peeling Process: The peeling based decoder we employ consists of finding a singleton bin, then identifying the single non-zero variable node connected to the bin, decoding its value and removing (*peeling off*) it's contribution from all the bin nodes connected to that variable node. The main idea behind this decoding scheme is that (for appropriately chosen parameters), at each iteration, peeling a singleton node off will induce at least one more singleton bin and the process of peeling off can be repeated iteratively. Although the main idea is similar for exact matching and the approximate matching scenarios, there are some subtle differences in their implementation.

Exact Matching: In the case of exact matching, we remove the decoded variable node's contribution from all the bin nodes it is connected to.

Approximate Matching: In this case, similar to the approach in [38], we remove the decoded variable node's contribution only from bins that are originally a singleton or a double-ton. We do not alter the bins which are classified to be multi-tons with degree more than two.

Note: The differences in peeling implementation for exact and approximate matching cases is because unlike exact matching the approximate matching may result in non-zero error (e_1) between the actual correlation value $r[k]$ and the estimate $\hat{r}[\hat{k}]$, i.e. $e_1 = |r[k] - \hat{r}[\hat{k}]|$ with $0 \leq e_1 \leq \eta M$. This may lead to error propagation if we use the same decoding scheme as in exact matching. Hence to overcome this problem we impose a constraint on the type of bin nodes participating in the peeling process.

We present the overall recovery algorithm, which comprises of *bin classification, singleton decoding* and *peeling process*, in the Algorithm.1 pseudo-code. Note that $\mathcal{N}(k)$ denote the neighborhood for variable node k i.e., the set of bins connected to k^{th} variable node.

Algorithm 1 Peeling based recovery algorithm

Compute $\widehat{\mathcal{H}}_{i,j}$ for $i \in [d], j \in [f_i]$. (See Eqn. (5.7))

while $\exists\ i, j : \widehat{\mathcal{H}}_{i,j} = \mathcal{H}_s$, **do**

$(\hat{k}, \hat{r}[\hat{k}]) =$ **Singleton-Decoder**$(\vec{\mathrm{z}}_{i,j})$

Assign $\hat{r}[\hat{k}]$ to $\hat{k}^{\text{th}}$ variable node

for $(i_0, j_0) \in \mathcal{N}(\hat{k})$ **do**

if *Exact Matching* ***then***

$\vec{\mathrm{z}}_{i_0,j_0} \leftarrow \vec{\mathrm{z}}_{i_0,j_0} - \hat{r}[\hat{k}]\vec{\mathrm{w}}^{\hat{k}}$

else

$\vec{\mathrm{z}}_{i_0,j_0} \leftarrow \vec{\mathrm{z}}_{i_0,j_0} - \hat{r}[\hat{k}]\vec{\mathrm{w}}^{\hat{k}}$ *only if* $\widehat{\mathcal{H}}_{i_0,j_0} = \mathcal{H}_s$ *or* $\mathcal{H}_d$

Re-do the bin classification for (i_0, j_0) *and compute* $\widehat{\mathcal{H}}_{i_0,j_0}$

Algorithm 2 Singleton-Decoder

Input: $\vec{\mathrm{z}}_{i,j}$

Output: $(\hat{k}, \hat{r}[\hat{k}])$

Estimate singleton index to be $\hat{k} = \underset{k \in \{j+lf_i\}}{\arg\max}\, \vec{\mathrm{z}}_{i,j}^{\dagger} \vec{\mathrm{w}}^k$

Estimate the value to be:

$$\hat{r}[\hat{k}] = \begin{cases} M & \text{Exact Matching case} \\ M - K & \text{Approximate Matching case} \end{cases}$$

Choosing f_i and α for various μ

For a given value of μ, we will describe how to choose the parameters d and f_i. Find a factorization for signal length $N = \prod_{i=0}^{d-1} P_i$ such that the set of integers $\{P_0, P_1, \ldots, P_{d-1}\}$ are pairwise co-prime and all the P_i are approximately equal. More precisely, let $P_i = \mathbf{F} + O(1)\ \forall i$ for some value $\mathbf{F}$. We can add zeros at the end of the vector $\vec{\mathrm{x}}$ and increase the length of the vector until we are able to find a factorization that satisfies this property.

- For $\mu < 0.5$: Choose $f_i = N/P_i$.

  Exact Matching: Find $d \in \mathbb{N} \backslash \{1, 2\}$ such that $\mu \in (\frac{1}{d}, \frac{1}{d-1}]$

  Approximate Matching: If $\mu \in (\frac{1}{8}, \frac{1}{2})$, choose $d = 8$. Else find $d \geq 8$ such that $\mu \in (\frac{1}{d}, \frac{1}{d-1})$

- For $\mu > 0.5$: Choose $f_i = P_i$

  Exact Matching: Find $d \in \mathbb{N} \backslash \{1, 2\}$ such that $\mu \in (1 - \frac{1}{d-1}, 1 - \frac{1}{d}]$

  Approximate Matching: If $\mu \in (\frac{1}{2}, \frac{1}{8})$, choose $d = 8$. Else find $d \geq 8$ such that

$$\mu \in (1 - \tfrac{1}{d-1}, 1 - \tfrac{1}{d})$$

Thus, for both the exact and approximate matching cases, for any $0 < \mu < 1$, we choose the down-sampling factors f_i to be approximately equal to N^α where $\alpha > 1 - \mu$.

Distributed Processing Framework

Given a database(or string) of length N, we divide the database into $G = N^\gamma$ blocks each of length $\tilde{N} = N/G$. Now each block can be processed independently (in parallel) using the RSIDFT framework with the new database length reduced from N to $\tilde{N}$. This distributed framework has the following advantages

- Firstly, this enables parallel computing and hence can be distributed across different workstations.
- Improves the sample and computational complexity by a constant factor.
- Sketch of the database needs to be computed only for a smaller block length and hence requires computation of only a shorter $\tilde{N}$ point FFT.
- Helps overcome implementation issues with memory and precision as the scale of the problem is reduced.

Sketches of $\vec{X}$ and $\vec{Y}$

As we have already seen in Section 5.4.1 the RSIDFT framework requires the values of $\vec{R}(= \vec{X} \odot \vec{Y})$ only at indices $\mathcal{S}$ or in other words we need $\vec{X}$ and $\vec{Y}$ only at the indices in set $\mathcal{S}$ of cardinality dBf_i. We assume that the sketch of $\vec{\mathrm{x}}$, $\vec{X}[\mathcal{S}] = \{X[i], i \in \mathcal{S}\}$ is pre-computed and stored in a database.

Computing the Sketch of $\vec{\mathrm{y}}$

For every new query $\vec{\mathrm{y}}$, only $\{\vec{Y}'[\mathcal{S}_{i,j}],\ i \in [d], j \in [B]\}$ needs to be computed where the subsets $\mathcal{S}_{i,j}$, defined in Eq. (5.4), are

$$\mathcal{S}_{i,j} := \{\ s_j + kg_i : k \in [f_i]\}, \quad i \in [d], j \in [B] \tag{5.8}$$

of cardinality f_i. Naively, the FFT algorithm can be used to compute N-pt DFT of $\vec{Y}'$ and the required subset of coefficients can be taken but this is inefficient and would be of $O(N \log N)$ complexity. Instead, we observe that $\vec{Y}'[\mathcal{S}_{i,j}]$ is $\vec{Y}'$ shifted by s_j and sub-sampled by a factor of g_i. Thus for a given (i, j) this corresponds to, in time domain, a point-wise multiplication by $[1, w_{s_j}, w_{s_j}^2, \ldots, w_{s_j}^{N-1}]$ followed by *folding* the signal into $g_i(= \frac{N}{f_i})$ signals each of length f_i and adding them up resulting in a single length-f_i signal denoted by $\vec{\mathrm{y}}'_{i,j}$. Formally the *folding* operation can be described as follows:

$$\vec{\mathrm{y}}'_{i,j} = \sum_{m=0}^{g_i-1} \vec{\mathrm{y}}'[mf_i : (m+1)f_i - 1] \odot [w_{s_j}^{mf_i}, w_{s_j}^{mf_i+1}, \ldots, w_{s_j}^{(m+1)f_i-1}],$$

where, $w_{s_j} = e^{-\frac{j2\pi s_j}{N}}$. Taking f_i-point DFT of $\vec{\mathrm{y}}'_{i,j}$ produces $\vec{Y}'[\mathcal{S}_{i,j}]$ i.e.

$$\vec{Y}'[\mathcal{S}_{i,j}] = \mathcal{F}_{f_i}\{\vec{\mathrm{y}}'_{i,j}\}$$

which is what we need in branch (i, j). To obtain all the samples in $\mathcal{S}$ required for the RSDIFT framework, the *folding* technique followed by a DFT needs to be carried out for each (i, j), for $i \in [d], j \in [B]$, a total of dB times N^α-point DFT.

The following theorem summarizes our main results.

Theorem 29. *Assume that a sketch of $\vec{\mathrm{x}}$ of size $O(\frac{N}{M} \log N)$ can be computed and*

stored. Then for the exact pattern matching and approximate pattern matching (with $K = \eta M$) problems, with the number of matches L scaling as $O(N^{\lambda})$, our algorithm has

- *a sketching function for $\vec{y}$ that computes $O(\frac{N}{M}\log N) \;=\; O\left(N^{1-\mu}\log N\right)$ samples*
- *a computational complexity of $O\left(\max\{N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N\}\right)$*
- *a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically in N*

When $L < O\left(\frac{N}{M}\right)$ (i.e. $\lambda < 1-\mu$), which is typically the interesting case, our algorithm has a sub-linear time complexity.

Proof. Section 5.7.1 and Section 5.7.2 analyzes the sample and computational complexity. Theorem 34 provides the proof for decaying error probability. □

5.5 Performance Analysis

In this section, we will analyze the overall probability of error involved in finding the correct matching positions. This can be done by analyzing the following three error events independently and then using a union bound to bound the total probability of error.

- $\mathcal{E}_1$-*Bin Classification*: Event that a bin is wrongly classified.
- $\mathcal{E}_2$-*Position Identification*: Given a bin is correctly identified as a singleton, event that the position of singleton is identified incorrectly.
- $\mathcal{E}_3$-*Peeling Process*: Given the classification of all the bins and the position identification of singletons in each iteration is accurate, event that the peeling process fails to recover the L significant correlation coefficients.

5.5.1 Bin Classification

Lemma 30. *The probability of bin classification error at any bin (i, j) can be upper bounded by*

$$\mathbb{P}[\mathcal{E}_1] \leq 6e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}}$$

Proof.

$$\begin{aligned}
\mathbb{P}[\mathcal{E}_1] = &\leq \mathbb{P}[\mathcal{E}_1|\widehat{\mathcal{H}}_{i,j} = \mathcal{H}_z] \;+\; \mathbb{P}[\mathcal{E}_1|\widehat{\mathcal{H}}_{i,j} = \mathcal{H}_s] \;+\; \mathbb{P}[\mathcal{E}_1|\widehat{\mathcal{H}}_{i,j} = \mathcal{H}_d \cup \mathcal{H}_m] \\
&\leq e^{-\frac{N^{\mu-\alpha}(1-2\eta)^2}{8}} + 2e^{-\frac{N^{\mu-\alpha}(1-4\eta)^2}{16}} + 2e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}} + e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}} \\
&\leq 6e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}}
\end{aligned}$$

where the inequalities in the third line are due to Lemmas 37, 38, 39 and 40 respectively provided in Appendix 5.6.2. □

5.5.2 Position Identification

Lemma 31. *Given that a bin (i, j) is correctly classified as a singleton, the probability of error in identifying the position of the non-zero variable node can be upper bounded by*

$$\mathbb{P}[\mathcal{E}_2] \leq \exp\left\{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2(c_1^2-1)}{8(c_1^2+1)}\right\} \;+\; \exp\left\{-\frac{N^{\mu+\alpha-1}\;(c_1(1-2\eta)-1)^2}{8(1+c_1^2)}\right\}$$

Proof. The detailed proof is provided in Section 5.6.3. □

5.5.3 Peeling Process

d	2	3	4	5	6	7	8
δ	1.000	0.4073	0.3237	0.2850	0.2616	0.2456	0.2336
$d\delta$	2.000	1.2219	1.2948	1.4250	1.5696	1.7192	1.8688

Table 5.2: Constants for various error floor values

To analyze the peeling process alone independently, we refer to a *oracle based peeling decoder* which has the accurate classification of all the bins and can accurately identify the position of the singleton given a singleton bin at any iteration. In other words, oracle based peeling decoder is the peeling part of our decoding scheme but with the assumption that the bin classification and position identification are carried out without any error.

Lemma 32 (Exact Matching)**.** *For the exact matching case, choose* $F^{d-1} = \delta N^{\alpha}$ *where* δ *is chosen as given in Table. 5.2. Then the oracle based peeling decoder:*

- *successfully uncovers all the* L *matching positions if* $L = \Omega(N^{\alpha})$ *and* $L \leq N^{\alpha}$*, with probability at least* $1 - O(1/N^{\frac{1}{d}})$
- *successfully uncovers all the* L *matching positions, if* $L = o(N^{\alpha})$*, with probability at least* $1 - e^{-\beta\varepsilon_1^2 N^{\alpha/(4l+1)}}$ *for some constants* $\beta, \varepsilon_1 > 0$ *and* $l > 0$.

Proof. We borrow this result from Pawar and Ramchandran's [20]. Although our RSDIFT framework and their robust-FFAST scheme have three main differences:

- We are computing smaller IDFT's to recover a sparse bigger IDFT whereas in [20] the same is true for DFT instead of IDFT.

- Our problem model is such that the sparse components of the signal space has only positive amplitude and thus our bin processing part (bin classification and position identification) is different when compared to [20].

- The sparsity of the signal L to be recovered is exactly known in the case of [20] whereas we have no information about L not even the order with which the quantity scales in N.

Irrespective of these differences, the Tanner graph representation of the framework and the peeling part of the decoder are identical to that of the robust-FFAST scheme. And thus the limit of the *oracle based peeling decoder* for our scheme is identical to that in the robust-FFAST scheme [20]. With respect to the third difference, in robust-FFAST scheme the authors choose $F^{d-1} = \delta k$ where k is the sparsity of the signal (which is assumed to be known) and show the first assertion of the lemma. They also showed that upto a constant fraction $(1 - \varepsilon)$ of k-variables node can be recovered with probability of failure that decays exponentially in N. In our case, since $L = o(N^{\alpha})$, this result translates to recovering all the L non-zero variable nodes with an exponentially decaying failure probability. □

In any iteration, given a singleton bin, the peeling process, in the case of approximate matching, runs the Singleton-Decoder algorithm on the bin only if it was either originally a singleton or originally a double-ton with one of the variable nodes being peeled off already. This is in contrast to the exact matching case where the peeling decoder runs the Singleton-Decoder on the bin irrespective of it's original degree. Hence we need to analyze the oracle based peeling decoder for the approximate

matching case separately compared to the exact matching case.

Lemma 33 (Approximate Matching)**.** *For the approximate matching case, choose parameter $d \geq 8$ as described in Sec. 5.4.1 and $F^{d-1} = 0.7663N^{\alpha}$. Then the oracle based peeling decoder:*

- *successfully uncovers all but a small fraction $\varepsilon = 10^{-3}$ of the L matching positions, if $L = \Omega(N^{\alpha})$ and $L \leq N^{\alpha}$ with a failure probability that decays exponentially in N*
- *successfully uncovers all the L matching positions, if $L = o(N^{\alpha})$, with probability at least $1 - e^{-\beta\varepsilon_1^2 N^{\alpha/(4l+1)}}$ for some constants $\beta, \varepsilon_1 > 0$ and $l > 0$.*

Proof. As mentioned earlier, the key difference in the approximate matching case is peeling off variable nodes from only singleton and double-tons. An identical peeling decoder is used and analyzed in the problem of group testing [38] by Lee, Pedarsani and Ramchandran which the authors refer to as SAFFRON scheme. In SAFFRON, the authors claim that for a graph ensemble which has a regular degree of d on the variable nodes and a Poisson degree distribution on the bins, this peeling decoder with a left degree of $d = 8$ and a total number of bins at least equal to $6.13k$ recovers at least $(1 - \varepsilon)$ fraction of the k non-zero variable nodes with exponentially decaying probability. Note that $\frac{6.13}{8} \approx 0.7663$ is approximately the number of bins per stage for $d = 8$. We also leverage the result from [20] that the Tanner graph representation of the robust-FFAST (or equivaently RSDIFT framework) has a Poisson degree distribution on the bins. Combining these two results gives us the required results. □

Theorem 34 (Overall Probability of Error)**.** *The RSIDFT framework succeeds with high probability asymptotically in N if the number of samples in each branch $f_i =$*

$N^\alpha + O(1)$ *satisfies the condition* $\alpha > 1 - \mu$.

Proof. The overall probability of error $\mathbb{P}[\mathcal{E}_{\text{total}}]$ can be bounded using an union bound on the three error events $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ given by

$$\mathbb{P}[\mathcal{E}_{\text{total}}] \;\leq\; \mathbb{P}[\mathcal{E}_1] + \mathbb{P}[\mathcal{E}_2] + \mathbb{P}[\mathcal{E}_3]$$

Using the expressions for error probabilities from Lemmas 30, 31, 32 and 33, we can see that all the terms vanish to zero as $N \to \infty$ if $\mu + \alpha - 1 \geq 0$, i.e. $\alpha \geq 1 - \mu$. □

5.6 Error Analysis

5.6.1 Chernoff Bounds

Lemma 35 (Hoeffding tail bound)**.** *Let* $X_1, X_2, \ldots, X_n$ *be a sequence of independent random variables such that* X_i *has mean* μ_i *and sub-Gaussian parameter* σ_i*. Then for any* $\delta > 0$*:*

$$\textbf{\textit{Upper Tail}}: \;\; \mathbb{P}\left[\sum (X_i - \mu_i) \geq \delta\right] \leq \exp\left\{-\frac{\delta^2}{2\sum \sigma_i^2}\right\}$$
$$\textbf{\textit{Lower Tail}}: \;\; \mathbb{P}\left[\sum (X_i - \mu_i) \leq -\delta\right] \leq \exp\left\{-\frac{\delta^2}{2\sum \sigma_i^2}\right\}$$

Note that for bounded random variables $X_i \in [a, b]$ *the sub-Gaussian parameter is* $\sigma_i = \frac{b-a}{2}$ *whereupon the upper tail Hoeffding bound can be simplified to*

$$\mathbb{P}\left[\sum_{i=1}^{n} (X_i - \mu_i) \geq \delta\right] \leq \exp\left\{-\frac{2\delta^2}{n(b-a)^2}\right\}. \tag{5.9}$$

Similarly the lower tail bound can be simplified.

Lemma 36 (Tail bounds for noise terms)**.** *Let us consider* $r[\theta_0], r[\theta_1], \ldots, r[\theta_{g_i-1}]$ *where* $\theta_j = \theta_0 + jf_i$ *and* $\theta_j \notin \{\tau_1, \ldots, \tau_L\}$ *is not one of the matching positions for any* j*. Then for any* $\delta > 0$*:*

Upper Tail:

$$\mathbb{P}\left[\left(\frac{1}{M}\sum_{j\in[g_i]}\sum_{k\in[M]} x[\theta_j+k]y[k]\right) \geq \delta\right] \leq \exp\left\{-\frac{M\delta^2}{2g_i}\right\}$$

Lower Tail:

$$\mathbb{P}\left[\left(\frac{1}{M}\sum_{j\in[g_i]}\sum_{k\in[M]} x[\theta_j+k]y[k]\right) \geq \delta\right] \leq \exp\left\{-\frac{M\delta^2}{2g_i}\right\}$$

Recall that $[g_i]$ *is used to denote the set* $\{0, 1, \ldots, g_i - 1\}$*.*

Proof. Since θ_j is not one of the matching positions for any j, $x[\theta_j + k] \neq y[k]$ and more importantly $x[\theta_j + k] \perp\!\!\!\perp y[k]\ \forall j, k$. This implies that $x[\theta_j + k]y[k] = \pm 1$ with equal probability and $\mathbb{E}[x[\theta_i + k]y[k]] = 0$. Let the set of random variables corresponding to a position θ_j be $S_j := \{x[\theta_j + k]y[k], k \in [M]\}$. It is clear that the random variables in the set S_j are all independent with respect to each other due to our i.i.d assumption on the database $\vec{\mathrm{x}}$ and θ_j being a non-matching position. For the case of $\mu < \alpha$ we have $M < f_i$ for large enough N thus resulting in non-overlapping parts of $\vec{\mathrm{x}}$ participating in the correlation coefficients $r[\theta_i]$ and $r[\theta_j]$. Hence it can be shown that $S_i \perp\!\!\!\perp S_j\ \ \forall i, j$ and we can apply the bounds from Eqn. (5.9) achieve the required result.

For the case of $\mu \geq \alpha$, $M > f_i$ for large enough N which results in a coefficient

$x[j]$ participating in multiple correlation coefficients $r[\theta_j]$. Therefore we pursue an alternate method of proof by defining

$$p_{j,l} := x[\theta_j + l] \sum_{k \in [\frac{M}{f_i}]} y[l + kf_i] \text{ for } l \in [f_i],$$

where $\theta_j = \theta_0 + jf_i$. W.L.O.G we assume that f_i divides M evenly although the proof can be extended on similar lines for the case where f_i does not divide M evenly. Now we can show that the required sum

$$\sum_{j \in [g_i]} \sum_{l \in [M]} x[\theta_j + l] y[l] = \sum_{j \in [g_i]} \sum_{l \in [f_i]} p_{j,l}.$$

From the above equivalent representation of the required sum, we need the following to achieve the required result:

- $p_{j,l}$ is sub-Gaussian with parameter $\sigma_i = \sqrt[\text{fft}]{\frac{M}{f_i}}$ since, from Eqn. (5.9),

$$\mathbb{P}\left[p_{j,l} \leq \delta\right] \leq \exp\left\{\frac{-\delta^2}{2\frac{M}{f_i}}\right\}$$

- The random variables $\{p_{j,l},\ \forall j, l\}$ are independent of each other due to the i.i.d assumption on the database

Now we can apply the tail bounds for sum of $g_i f_i$ sub-Gaussian random variables each with sub-Gaussian parameter $\sqrt{\frac{M}{f_i}}$ from Lemma 35 and arrive at the required result. □

5.6.2 Bin Classification Errors

We employ classification rules based only on the first element of the measurement vector at bin (i, j) which can be given by

$$Z[1] = \begin{cases} \sum_{\ell=0}^{g_i-1} \sum_{k=0}^{M-1} n_{l,k} & \text{if} \quad \mathcal{H} = \mathcal{H}_z \\ M_1 + \sum_{\ell=0}^{g_i-2} \sum_{k=0}^{M-1} n_{l,k} & \text{if} \quad \mathcal{H} = \mathcal{H}_s \\ M_1 + M_2 + \sum_{\ell=0}^{g_i-3} \sum_{k=0}^{M-1} n_{l,k} & \text{if} \quad \mathcal{H} = \mathcal{H}_d \end{cases} \tag{5.10}$$

where $n_{l,k} = x[\theta_\ell + k]y[k]$ and $\theta_\ell \notin \{\tau_1, \tau_2, \ldots, \tau_L\}$. Also for the case of exact matching $M_1 = M_2 = M$ whereas in the case of approximate matching the values of $M_1, M_2 \in [M(1-2\eta) : M]$.

Lemma 37 (zero-ton)**.** *Given that the bin (i, j) is a zero-ton, the classification error can be bounded by*

$$\mathbb{P}[\mathcal{E}_1|\mathcal{H}_z] \leq e^{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2}{8}}$$

Proof. The above expression can be derived by observing that a bin is not classified as zero-ton if $\frac{Z[1]}{M} \geq \frac{1-2\eta}{2}$. Let us denote the probability of this event as p_{z1} which can be bounded as:

$$\begin{aligned} p_{z1} =& \mathbb{P}\left[\frac{Z[1]}{M} \geq \frac{1-2\eta}{2}\right] \\ &\leq e^{-\frac{Mg_i(1-2\eta)^2}{8g_i^2}} \\ &\approx e^{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2}{8}} \end{aligned}$$

where the second bound is due to Eqn. (5.10) and Lemma 36. The approximation

in the third line is from our design that all the g_i are chosen such that $g_i \approx N^{1-\alpha}$ and $M = N^{\mu}$. □

Lemma 38 (singleton)**.** *Given that the bin (i, j) is a singleton, the classification error can be bounded by*

$$\mathbb{P}[\mathcal{E}_1|\mathcal{H}_s] \leq 2e^{-\frac{N^{\mu+\alpha-1}(1-4\eta)^2}{16}}$$

Proof. We observe that a bin is not classified as singleton if $\frac{Z[1]}{M} \leq \frac{1-2\eta}{2}$ or $\frac{Z[1]}{M} \geq \frac{3-4\eta}{2}$. Let us denote the probability of the two events as p_{s1} and p_{s2} respectively which can be bounded as:

$$\begin{aligned}
p_{s1} &= \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-2}\sum_{k=0}^{M-1} n_{l,k} \leq \frac{1-2\eta}{2} - \frac{M_1}{M}\right] \\
&\leq \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-2}\sum_{k=0}^{M-1} n_{l,k} \leq -\frac{1-2\eta}{2}\right] \\
&\leq e^{-\frac{Mg_i(1-2\eta)^2}{16g_i^2}} \\
&\approx e^{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2}{16}}
\end{aligned}$$

where we used *lower tail* of Lemma 35 and $g_i \approx N^{1-\alpha}$ and the lower bound on $\frac{M_1 \text{ffl}}{M} \geq (1-2\eta)$. Similarly p_{s2} can be upper bounded by:

$$\begin{aligned}
p_{s2} &= \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-2}\sum_{k=0}^{M-1} n_{l,k} \geq \frac{3-4\eta}{2} - \frac{M_1}{M}\right] \\
&\leq \mathbb{P}\left[\frac{1}{Mg_i}\sum_{\ell=0}^{g_i-2}\sum_{k=0}^{M-1} n_{l,k} \geq -\frac{1-4\eta}{2g_i}\right] \\
&\approx e^{-\frac{N^{\mu+\alpha-1}(1-4\eta)^2}{8}}
\end{aligned}$$

Thus the overall probability of error for classifying a singleton can be obtained by

combining p_{s1} and p_{s2}. $\square$

Lemma 39 (double-ton)**.** *Given that the bin (i, j) is a double-ton, the classification error can be bounded by*

$$\mathbb{P}[\mathcal{E}_1|\mathcal{H}_d] \leq 2e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}}$$

Proof. We observe that a bin is not classified as double-ton if $\frac{Z[1]}{M} \leq \frac{3-4\eta}{2}$ or $\frac{Z[1]}{M} \geq \frac{5-6\eta}{2}$. Let us denote the probability of these two events as p_{d1} and p_{d2} respectively which can be bounded similar to Lemma 38.

$$\begin{aligned}
p_{d1} &= \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-3}\sum_{k=0}^{M-1} n_{l,k} \leq \frac{3-4\eta}{2} - \frac{M_1+M_2}{M}\right] \\
&\leq \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-3}\sum_{k=0}^{M-1} n_{l,k} \leq -\frac{1-4\eta}{2}\right] \\
&\leq e^{-\frac{M(g_i-2)(1-4\eta)^2}{16(g_i-2)^2}} \\
&\approx e^{-\frac{N^{\mu+\alpha-1}(1-4\eta)^2}{16}}.
\end{aligned}$$

Similarly p_{d2} can be bounded as

$$\begin{aligned}
p_{d2} &= \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-3}\sum_{k=0}^{M-1} n_{l,k} \geq \frac{5-6\eta}{2} - \frac{M_1+M_2}{M}\right] \\
&\leq \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-3}\sum_{k=0}^{M-1} n_{l,k} \geq \frac{1-6\eta}{2}\right] \\
&\leq e^{-\frac{M(g_i-2)(1-6\eta)^2}{8(g_i-2)^2}} \\
&\approx e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{8}}
\end{aligned}$$

where we use the lower bounds $M_1, M_2 \leq M$. $\square$

Lemma 40 (multi-ton)**.** *Given that the bin (i, j) is a multi-ton, the classification error can be bounded by*

$$\mathbb{P}[\mathcal{E}_1|\mathcal{H}_m] \leq e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}}$$

Proof. We observe that a bin is not classified as multi-ton if $\frac{Z[1]}{M} \leq \frac{5-6\eta}{2}$. Let us denote the probability of this event as p_{m1} which can be bounded as:

$$\begin{aligned}
p_{m1} &= \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-3}\sum_{k=0}^{M-1} n_{l,k} \leq \frac{5-6\eta}{2} - \frac{M_m}{M}\right] \\
&\leq \mathbb{P}\left[\frac{1}{M}\sum_{\ell=0}^{g_i-m}\sum_{k=0}^{M-1} n_{l,k} \leq -\frac{1-6\eta}{2}\right] \\
&\leq e^{-\frac{M(g_i-m)(1-4\eta)^2}{16(g_i-m)^2}} \\
&\leq e^{-\frac{M(1-6\eta)^2}{16n_i}} \\
&\approx e^{-\frac{N^{\mu+\alpha-1}(1-6\eta)^2}{16}}.
\end{aligned}$$

□

5.6.3 Position Identification

We will analyze the singleton identification in two separate cases:

- $\mathcal{E}_{21}$: Event where the position is identified incorrectly when the bin is classified correctly a singleton
- $\mathcal{E}_{22}$: In the case of approximate matching, event where the position is identified incorrectly when the bin is originally a double-ton and one of the non-zero variable nodes has already been peeled off

Definition 41 (Mutual Incoherence)**.** *The mutual incoherence $\mu_{max}(\mathbf{W})$ of a matrix $\mathbf{W} = [\vec{\mathrm{w}}_1\ \vec{\mathrm{w}}_2\ \cdots \vec{\mathrm{w}}_i \cdots \vec{\mathrm{w}}_N]$ is defined as*

$$\mu_{max}(\mathbf{W}) \triangleq \max_{\forall i \neq j} \frac{|\vec{\mathrm{w}}_i^{\dagger} \vec{\mathrm{w}}_j|}{||\vec{\mathrm{w}}_i||.||\vec{\mathrm{w}}_j||}$$

Lemma 42 (Mutual Incoherence Bound for sub-sampled IDFT matrix [[20], Proposition A.1)**.** *The mutual incoherence $\mu_{max}(\mathbf{W_{i,k}})$ of the sensing matrix $\mathbf{W}_{i,k}$ (defined in Equation 5.6), with B shifts, is upper bounded by*

$$\mu_{max} < 2\sqrt{\frac{\log(5N)}{B}}$$

Proof. The proof follows similar lines as the proof for Lemma V.3. in [20]. □

Lemma 43. *For some constant $c_1 \in \mathbb{R}$ and the choice of $B = 4c_1^2 \log 5N$, the probability of error in identifying the position of a singleton at any bin (i, j) can be upper bounded by*

$$\mathbb{P}[\mathcal{E}_{21}] \leq \exp\left\{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2(c_1^2-1)}{8(c_1^2+1)}\right\}$$

Proof. Let j_p be the variable node participating in the singleton (i, j). Then the observation vector $\vec{\mathrm{z}}_{i,j}$ is given by

$$\underline{z}_{i,j} = \begin{bmatrix} \vec{\mathrm{w}}_{j_1}, \vec{\mathrm{w}}_{j_2}, & \cdots & \vec{\mathrm{w}}_{j_p}, & \cdots & \vec{\mathrm{w}}_{j_{g_i}} \end{bmatrix} \times \begin{bmatrix} n_1 \\ \vdots \\ r[j_p] \\ \vdots \\ n_j \\ \vdots \\ n_{g_i} \end{bmatrix}$$

$$= r[j_p]\ \vec{\mathrm{w}}_{j_p} + \sum_{k \neq p} n_k \vec{\mathrm{w}}_{j_k}$$

where for convenience we use a simpler notation $j_k = j + (k-1)\frac{N}{f_i}, \vec{\mathrm{w}}_{j_k} = \vec{\mathrm{w}}^{j_k}$ as defined in Equation and $n_l = \sum_{k=0}^{M-1} x[\theta_\ell + k]y[k]$ as defined in Equation (5.10).

The estimated position $\hat{p}$ is given by

$$\hat{p} = \arg\max_{l} \ \frac{\vec{\mathrm{w}}_{j_l}^{\dagger} \underline{z}_{i,j}}{B} \tag{5.11}$$

where $\dagger$ denotes the conjugate transpose of the vector. Also note that $||\vec{\mathrm{w}}_{j_k}|| = B$ for any j and k. From Equation (5.11) we observe that the position is wrongly identified when $\exists p'$ such that

$$r[j_p] + \frac{1}{B}\sum_{k \neq p} n_k \vec{\mathrm{w}}_{j_p}^{\dagger} \vec{\mathrm{w}}_{j_k} \leq \frac{r[j_p]}{B}\ \vec{\mathrm{w}}_{j_{p'}}^{\dagger} \vec{\mathrm{w}}_{j_p} + n_{p'} + \frac{1}{B}\sum_{k \neq p,p'} n_k \vec{\mathrm{w}}_{j_{p'}}^{\dagger} \vec{\mathrm{w}}_{j_k}$$

$$\leftrightarrow \sum_{k \neq p,p'} \alpha_k n_k + \beta n_{p'} \geq r[j_p]\left(1 - \frac{\vec{\mathrm{w}}_{j_{p'}}^{\dagger} \vec{\mathrm{w}}_{j_p}}{B}\right) \geq M(1-2\eta)(1-\mu_{\max})$$

where α_k and β are constants and can be shown to be in the range $\alpha_k \in [-2\mu_{\max}, 2\mu_{\max}]$ and $\beta \in [1-\mu_{\max}, 1+\mu_{\max}]$. Now using the bound given Chernoff Lemma in

Lemma 36 we obtain

$$\begin{aligned}
\mathbb{P}[\mathcal{E}_{21}] &\leq \exp\left\{-\frac{2M(1-2\eta)^2(1-\mu_{\max})^2}{16g_i\mu_{\max}^2+4(1+\mu_{\max})^2}\right\} \\
&\leq \exp\left\{-\frac{2M(1-2\eta)^2(1-\mu_{\max})^2}{16(g_i\mu_{\max}^2+1)}\right\} \\
&\leq \exp\left\{-\frac{2M(1-2\eta)^2(c_1-1)^2}{16(g_i+c_1^2)}\right\} \\
&\approx \exp\left\{-\frac{N^{\mu+\alpha-1}(1-2\eta)^2(c_1^2-1)}{8(c_1^2+1)}\right\}
\end{aligned}$$

The second inequality follows by the definition of $\mu_{\max} \leq 1$. We choose $B = 4c_1^2 \log 5N$, and substituting $\mu_{\max} \leq 2\sqrt{\frac{\log 5N}{B}} = 1/c_1$ (Lemma 42) we get the third inequality.

□

Lemma 44. *For some constant $c_1 \in \mathbb{R}$ and the choice of $B = 4c_1^2 \log 5N$, the probability of error in identifying the position of second non-zero variable node at a double-ton at any bin (i, j), given that the first position identification is correct, can be upper bounded by*

$$\mathbb{P}[\mathcal{E}_{22}] \leq \exp\left\{-\frac{N^{\mu+\alpha-1}\ (c_1(1-2\eta)-1)^2}{8(1+c_1^2)}\right\}$$

Proof. $\mathcal{E}_{22}$**:**

Let j_p and $j_{\tilde{p}}$ be the two variable nodes participating in the doubleton (i, j). Then the observation vector $\vec{z}_{i,j}$ is given by

$$\underline{z}_{i,j} = \left[\vec{\mathrm{w}}_{j_1}, \vec{\mathrm{w}}_{j_2}, \quad \cdots \quad \vec{\mathrm{w}}_{j_p}, \quad \cdots \quad \vec{\mathrm{w}}_{j_{g_i}}\right] \times \begin{bmatrix} n_1 \\ \vdots \\ r[j_p] \\ \vdots \\ n_j \\ \vdots \\ r[j_{\tilde{p}}] \\ \vdots \\ n_{g_i} \end{bmatrix}$$

$$= r[j_p]\ \vec{\mathrm{w}}_{j_p} + r[j_{\tilde{p}}]\ \vec{\mathrm{w}}_{j_{\tilde{p}}} + \sum_{k \neq p} n_k \vec{\mathrm{w}}_{j_k}$$

Let the contribution from $j_{\tilde{p}}$ be peeled off from the doubleton at some iteration, then we get

$$\vec{\mathrm{z}}_{i,j} = r[j_p]\ \vec{\mathrm{w}}_{j_p} + \frac{e_1}{B}\ \vec{\mathrm{w}}_{j_{\tilde{p}}} + \sum_{k \neq p} n_k \vec{\mathrm{w}}_{j_k}$$

where $e_1 \in [-\eta M, \eta M]$ is an extra error term induced due to peeling off.

Now the estimated second position $\hat{p}$ is calculated using Equation (5.11). We can observe that the position is wrongly identified when $\exists p'$ such that

$$\frac{\vec{\mathrm{w}}_{j_p}^{\dagger} \underline{z}_{i,j}}{B} \leq \frac{\vec{\mathrm{w}}_{j_{p'}}^{\dagger} \underline{z}_{i,j}}{B}$$

$$\implies r[j_p] + \frac{1}{B}\sum_{k\neq p,\tilde{p}} n_k \vec{\mathrm{w}}_{j_p}^{\dagger}\vec{\mathrm{w}}_{j_k} + \frac{e_1}{B}\vec{\mathrm{w}}_{j_p}^{\dagger}\vec{\mathrm{w}}_{j_{\tilde{p}}}$$
$$\leq \frac{r[j_p]}{B}\ \vec{\mathrm{w}}_{j_{p'}}^{\dagger}\vec{\mathrm{w}}_{j_p} + n_{p'} + \frac{1}{B}\sum_{k\neq p,p',\tilde{p}} n_k \vec{\mathrm{w}}_{j_{p'}}^{\dagger}\vec{\mathrm{w}}_{j_k} + \frac{e_1}{B}\vec{\mathrm{w}}_{j_{p'}}^{\dagger}\vec{\mathrm{w}}_{j_{\tilde{p}}}$$

$$\leftrightarrow \sum_{k\neq p,p',\tilde{p}} \alpha_k n_k + \beta n_{p'} \geq r[j_p]\left(1 - \frac{\vec{\mathrm{w}}_{j_{p'}}^{\dagger}\vec{\mathrm{w}}_{j_p}}{B}\right) - \frac{2\eta M}{B}\vec{\mathrm{w}}_{j_{p'}}^{\dagger}\vec{\mathrm{w}}_{j_{\tilde{p}}}$$

$$\geq M(1-2\eta)(1-\mu_{\max}) - 2\eta M\mu_{\max} = M(1-2\eta-\mu_{\max})$$

where α_k and β are constants and can be shown to be in the range $\alpha_k \in [-2\mu_{\max}, 2\mu_{\max}]$ and $\beta \in [1-\mu_{\max}, 1+\mu_{\max}]$. Now using the bound given by Chernoff Lemma in Lemma 36 we obtain

$$\begin{aligned}
\mathbb{P}[\mathcal{E}_{22}] &\leq \exp\left\{-\frac{2M(1-2\eta-\mu_{\max})^2}{16g_i\mu_{\max}^2 + 4(1+\mu_{\max})^2}\right\} \\
&\leq \exp\left\{-\frac{2M(1-2\eta-\mu_{\max})^2}{16(g_i\mu_{\max}^2+1)}\right\} \\
&\leq \exp\left\{-\frac{M(c_1(1-2\eta)-1)^2}{8(g_i+c_1^2)}\right\} \\
&\leq \exp\left\{-\frac{N^{\mu+\alpha-1}\ (c_1(1-2\eta)-1)^2}{8(1+c_1^2)}\right\}
\end{aligned}$$

where for the choice of $B = 4c_1^2\log 5N$, $\mu_{\max} \leq 2\sqrt{\frac{\log 5N}{B}} = 1/c_1$.

□

5.7 Complexity Analysis

In this section, we will analyze the sketching complexity which is the number of samples we access from the sketch of the signal $\vec{\mathrm{x}}$ stored in the database and the

computational complexity as a function of the system parameters.

5.7.1 Sample Complexity

In each branch of the RSDIFT framework we down-sample the N samples by a factor of $\frac{N}{f_i}$ to get $f_i \approx N^{\alpha}$ samples. We repeat this for a random shift in each branch for $B = O(\log N)$ branches in each stage thus resulting in a total of $O(N^{\alpha} \log N)$ samples per block per stage. We repeat this for $d = \frac{1}{1-\alpha}$ such stages resulting in a total of $dN^{\alpha} \log N$ samples per block. So, the total number of samples is given by

$$S = O\left(dN^{\alpha} \log N\right) = O(N^{1-\mu} \log N)$$

5.7.2 Computational Complexity

As described in Eq. 5.3, the computation of $\vec{\mathrm{r}}$ involves three steps:

1. Operation - I: Since we assume that the sketch of database $\vec{\mathrm{x}}$, $\mathcal{F}_N\{\vec{\mathrm{x}}\}$, is pre-computed, we do not include this in computational complexity.

2. Operation - II: As described in Section 5.4.1, in each branch (i, j), we use a folding based technique to compute the sketch of $\vec{\mathrm{y}}$, $\mathcal{F}_N\{\vec{\mathrm{y}}'\}$ at points in the set $\mathcal{S}_{i,j}$. The folding technique involves two steps: folding and adding (aliasing) which has a complexity of $O(M)$ computations , and computing f_i-point DFTs that takes $O(N^{\alpha} \log N^{\alpha})$ computations. So, for a total of dB branches the number of computations in this step is given by

$$\begin{aligned} C_{II} &= dB \left(\underbrace{N^{\mu}}_{\text{Folding}} + \underbrace{N^{\alpha} \log N^{\alpha}}_{\text{Shorter FFTs}} \right) \\ &= O(\max(N^{1-\mu} \log^2 N, N^{\mu} \log N)). \end{aligned}$$

Note: Folding and adding, for each shift, involves adding $N^{1-\alpha}$ vectors of length N^{α}. We know that the length of the query is $M = N^{\mu}$, i.e., the number of non-zero elements in $\vec{y}$ (zero-padded version of the query) is M and hence we only need to compute M additions instead of length of the vector N.

3. Operation - III: Computing $\mathcal{F}_N^{-1}\{\vec{x}\}$ involves two parts:

   RSIFT framework and the decoder. The RSIFT framework involves computing smaller f_i point IDFTs, which takes approximately $O(N^{\alpha}\log N^{\alpha})$ computations in each branch. For a total of dB branches, we get a complexity of $O(dBN^{\alpha}\log N^{\alpha})$. In the decoding process, the dominant computation is from position identification. Each position identification process involves correlating the observation vector of length B with $\frac{N}{f_i} \approx N^{1-\alpha}$ column vectors, which amounts to $BN^{1-\alpha}$ computations. There will be a maximum of dL such position identifications, which gives a complexity of$O(dLBN^{1-\alpha})$. Now, plugging in $\alpha = 1-\mu$ (condition for vanishing probability of error) the total number of computations involved in this step, C_I, is given by

$$\begin{aligned} C_{III} &= dB\ (\ \underbrace{O(N^{\alpha}\log N^{\alpha})}_{\text{Shorter IFFTs /block/stage}} + \underbrace{L\ N^{1-\alpha}}_{\text{Correlations}}\ ) \\ &= O(\max(N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N)). \end{aligned}$$

Thus, the total number of computations, $C = \max\{C_{II}, C_{III}\}$, is given by

$$C = O(\max\{N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N\})$$

5.8 Simulation Results

Simulations were carried out to test the performance of RSIDFT framework for exact matching scenario on a database of length $N = 10^{12}$ for two different query lengths $M = 10^5$ ($\mu = 0.41$) and $M = 10^3$ ($\mu = 0.25$). The database was generated as a equiprobable $\{+1, -1\}$ sequence of length N. A substring of length M from the generated database is presented as a query. Also the chosen query was repeated at $L = 10^6$ randomly chosen locations in the database. The sample gain, defined as the ratio of N to the number of samples used from the sketch of database, was varied and the probability of RSIDFT framework to miss a match (P_e), as defined below, was measured.

$$P_e = \frac{\#\text{ of incorrectly identified locations}}{L}$$

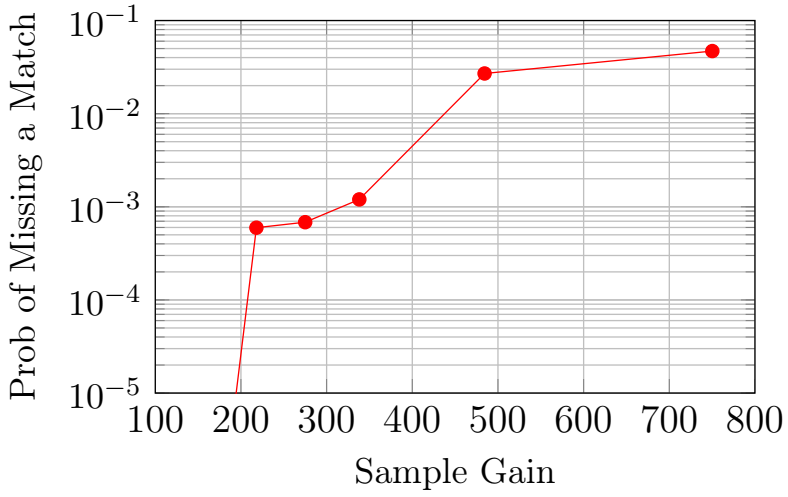

Figure 5.4: Plot of probability of error in a match vs. sample gain for exact matching of a query of length $M = 10^5(\mu = 0.41)$ from a equiprobable binary {+1,-1} sequence of length $N = 10^{12}$, divided into $G = 10^5$ blocks each of length $\tilde{N} = 10^6$. The substring was simulated to repeat in $L = 10^6(\lambda = 0.5)$ locations uniformly at random.

The plots of P_e vs. sample gain, is presented for two different query lengths: $M = 10^5$ ($\mu = 0.41$) in Figure 5.4 and $M = 10^3$ ($\mu = 0.25$) in Figure 5.5. As can be inferred from the plots we achieve a sample gain of 200-300 (depending on the tolerable error probability) for the query length corresponding to $\mu = 0.41$ and a sample gain of 2-4 for $\mu = 0.25$. This sample gain results from an average number of samples per branch $f_i \approx 9.25 \times 10^7$ ($\alpha = 0.66$) for $\mu = 0.41$, and $f_i \approx 6.94 \times 10^9$ ($\alpha = 0.82$) for $\mu = 0.25$. The trend in the results almost matches with the theoretical findings of $\alpha = 1 - \mu$. We also notice a sharp threshold in the sample gain, below which the RSIDFT framework succeeds with very high probability.

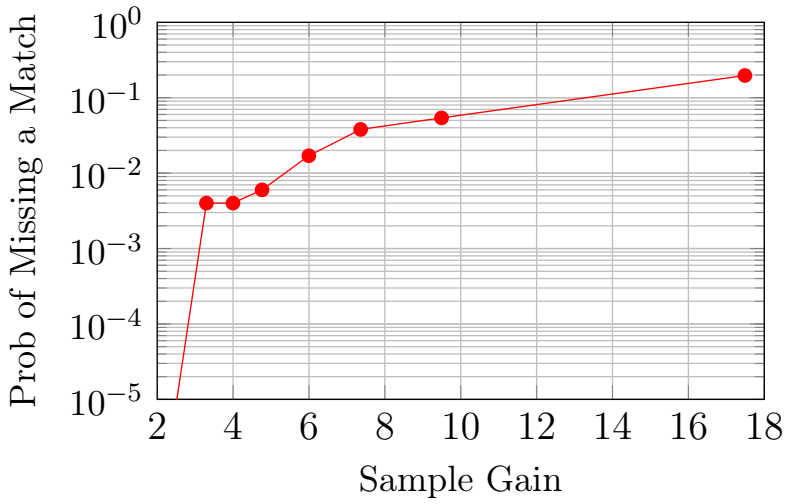

Figure 5.5: Plot of probability of error in a match vs. sample gain for exact matching of a query of length $M = 10^3(\mu = 0.25)$ from a equiprobable binary {+1,-1} sequence of length $N = 10^{12}$, divided into $G = 10^6$ blocks each of length $\tilde{N} = 10^6$. The substring was simulated to repeat in $L = 10^6(\lambda = 0.5)$ locations uniformly at random.

5.9 Extensions

5.9.1 Important Algorithmic Improvements

We consider the fundamental question: Can the computational complexity be decreased below $O(N^{\mu+\lambda}\log N)$, for $\mu > 0.5$? Intuitively, when μ is large, the ratio of the peak in the correlation to the off-peak values is very large and this should facilitate identifying the peak with lower space and computational complexity. This intuition holds true in the space complexity $O(N^{1-\mu})$ of our algorithm, but the computational complexity could still be improved. Here, we present a modification to our algorithm that allows for improvement in time complexity for $\mu > 0.5$, by trading for some space complexity.

The modification comes in the following parts of the original algorithm:

- **Sketch of database** $\vec{\mathrm{x}}$: For $\mu > 0.5$, before the sketching operation is performed, we down-sample $\vec{\mathrm{x}}$ by a factor of $D = O(N^{\delta})$, with $\delta = \mu - 0.5$, to obtain a smaller database $\vec{\mathrm{x}}_s$ of size $N' = O(N^{1-\delta}) = O(N^{1.5-\mu})$. The down-sampling operation is given by,

$$x_s[m] = \sum_{p=0}^{D-1} x[mD+p], \quad m = \{0, 1, \cdots, N'-1\} \tag{5.12}$$

  After this down-sampling step, we compute the sketch of $\vec{\mathrm{x}}_s$, $\vec{X}_s[\mathcal{S}] = \{X_s[i], i \in \mathcal{S}\}$, where the set $\mathcal{S} = \cup_{i,j}\mathcal{S}_{i,j}$ in Equation (5.4) is redefined for the reduced length N' using the new parameters $f_i' = O(N'^{\alpha}) = O(N^{(1-\delta)\alpha})$, $g_i' = O\left(N^{(1-\delta)(1-\alpha)}\right)$, $B = O(\log N')$.

- **Sketch of query** $\vec{\mathrm{y}}$: For $\mu > 0.5$, we also down-sample the query $\vec{\mathrm{y}}$ by $D = N^{\delta}$ to obtain $\vec{\mathrm{y}}_s$ of length $M' = M/D = O(N^{\mu-\delta})$, with $y_s[m] = y[mD], \quad m =$

$\{0, 1, \cdots, N' - 1\}$. The sketching operation for the query is similar to Section 5.4.1, where we replace $\vec{y}$ by the down-sampled version $\vec{y}_s$, and the new parameters $f_i' = O(N^{(1-\delta)\alpha})$, $g_i' = O(N(1-\delta)(1-\alpha)), B' = O(\log N')$ for f_i, g_i and B respectively.

- **RSIDFT algorithm**: This is similar to the Section 5.4.1 with one important change. Instead of $\vec{R}$, we input $\vec{R}_s = \vec{X}_s \odot \vec{Y}_s'$ to RSIDFT block with the new parameters being g_i', f_i', B'. Hence, the output of the peeling decoder is the correlation of the down-sampled vectors, denoted as $\vec{r}_s$, and is defined by,

$$r_s[m] = (\vec{x}_s * \vec{y}_s)[m] \triangleq \sum_{i=1}^{M'} y_s[i] x_s[m + i - 1]_{M'} \tag{5.13}$$

- **Position identification**: The RSIDFT algorithm identifies peaks in the correlation of the down-sampled vector $\vec{r}_s$ and not the actual correlation vector $\vec{r}$. Hence, we only get an estimate of the positions $\vec{\tau}$ up to an error of $\pm D$. In order to find the exact location, we need to perform matching in the vicinity $\vec{x}_{\hat{\tau}} = x[\hat{\tau}_s : (D-1)\hat{\tau}_s]$ of the estimate $\hat{\tau}_s$ to narrow down to the exact position $\hat{\tau}$. To perform this matching, we can either use RSIDFT framework again on the shorter length signal or compute naively by traversing through the vicinity array $\vec{x}_{\hat{\tau}}$ for all the L location estimates.

Note that the we follow the original algorithm for $\mu \leq 0.5$ and the extra down-sampling steps described above are performed only for $\mu > 0.5$.

The following theorem summarizes the results from the improved algorithm. Notice the increase in space complexity and decrease in time complexity for $\mu > 0.5$.

Theorem 45. *Assume that a sketch of $\vec{x}$ can be computed and stored. Then for the exact pattern matching and approximate pattern matching (with $K = \eta M$) problems,*

with the number of matches L scaling as $O(N^{\lambda})$, our algorithm has

- *a sketching function for $\vec{\mathrm{y}}$ that computes S samples given by*

$$S = \begin{cases} O\left(N^{1-\mu}\log N\right), & \forall \mu \leq 0.5 \\ O\left(\sqrt{N}\log N\right), & \forall \mu > 0.5 \end{cases} \tag{5.14}$$

.

- *a computational complexity of*

$$C = \begin{cases} O\left(\max\{N^{1-\mu}\log^2 N, N^{\mu+\lambda}\log N\}\right), & \forall \mu \leq 0.5 \\ O\left(N^{\max\{0.5,\ 1-\mu+\lambda\}}\log^2 N\right), & \forall \mu > 0.5 \end{cases} \tag{5.15}$$

.

- *a decoder that recovers all the L matching positions with a failure probability that approaches zero asymptotically in N*

When $L < O\left(\frac{N}{M}\right)$ (i.e. $\lambda < 1-\mu$), which is typically the interesting case, our algorithm has a sub-linear time complexity.

Proof. The proof follows similar structure as the proof in Theorem 29 with the new parameters $f_i' = (N')^{\alpha} = N^{(1-\delta)\alpha},\ g_i' = N^{(1-\delta)(1-\alpha)},\ B = O(\log N')$.

Sample complexity: Similar to Section 5.7.1, it can be shown that the sample complexity $S = O(dN^{1-\delta})$. For $\mu \leq 0.5$, $\delta = 0$, and hence, the complexity is same as the original algorithm. For $\mu > 0.5$, by construction, $\delta = \mu - 0.5$, giving a sample complexity $S = O\left(\sqrt{N}\log N\right)$.

Computational complexity: The complexity analysis is similar to Section 5.7.2, where we ignore the sketching complexity C_I of $\vec{\mathrm{x}}$, and the sketching complexity C_{II} of $\vec{\mathrm{y}}$ is now reduced due to the down-sampling operation. For the new set

of parameters $\{D, N', M', g', f'\}$, $C_{II} = O(\max(N^{1-\mu+\delta}\log^2 N, N^{\mu-\delta}\log N))$. Similarly the complexity of RSIDFT framework C_{III}, can be re-derived as $C_{III} = O(\max(N^{1-\mu+\delta}\log^2 N, N^{\mu+\lambda-2\delta}\log N))$. Substituting $\delta = \mu - 0.5$, we get the $C = \max\{C_{II}, C_{III}\} = O\left(N^{\max\{0.5,\ 1-\mu+\lambda\}}\log^2 N\right)$.

Error Analysis: The following lemma provides the overall probability of error (similar to Theorem34) for the improved algorithm.

Lemma 46 (Overall Probability of Error)**.** *The RSIDFT framework succeeds with high probability asymptotically in N if the number of samples in each branch $f_i' = N^{(1-\delta)\alpha} + O(1)$ satisfies the condition $\alpha \geq \frac{1-\mu+\delta}{1-\delta}$.*

Proof. The overall probability of error $\mathbb{P}[\mathcal{E}_{\text{total}}]$ can be bounded using an union bound on the three error events $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ given by

$$\mathbb{P}[\mathcal{E}_{\text{total}}] \ \leq \ \mathbb{P}[\mathcal{E}_1] + \mathbb{P}[\mathcal{E}_2] + \mathbb{P}[\mathcal{E}_3]$$

We can re-derive results similar to Lemmas 30, 31, 32 and 33, and see that all the terms vanish to zero as $N \to \infty$ if $\mu + \alpha - \delta - \alpha\delta - 1 \geq 0$, i.e. $\alpha \geq \frac{1-\mu+\delta}{1-\delta}$. □

□

Notice that the improved algorithm complexity saturates to $O(N^{0.5}\log^2 N)$. We can further decrease the complexity using two approaches. The first approach is to replace the sensing matrix $\mathbf{W}_{i,k}$ in Equation (5.5) by a structured set of rows of the DFT matrix and to use the fast and accurate single frequency estimator technique from [39], which was used for compressed sensing in [40]. The second approach is to improve our sub-sampling strategy of $\vec{\mathrm{x}}$ and $\vec{\mathrm{y}}$, which is currently not sampled at its maximum rate due to some inherent constraints of the algorithm. Both these

approaches trade off computational complexity for decreased robustness to the off-peak values and our goal is to analyze the tradeoff.

5.9.2 Real-World Data Sets and Correlations

While our preliminary results show substantial gains, they are derived from synthetic data, and they focus on the case when $\vec{x}$ is a sequence of i.i.d. binary digits. Real world data such as audio and images often exhibit correlation in time or space; this raises the question: Can the RSIDFT algorithm provide substantial gains for real-world data?

5.9.2.1 Non-binary Data

First of all, our proposed technique has a substantial advantage over the Burrows-Wheeler transform (BWT) based approaches in handling non-binary (or, real-valued) data since the complexity of our algorithm is independent of the cardinality of the alphabet $\mathcal{A}$, whereas the complexity of approximate matching based on BWT grows with $|\mathcal{A}|$, and grows exponentially with K. Even in the case of exact matching, the sketching complexity of the BWT based approach grows with the cardinality of $\mathcal{A}$. So, our algorithm naturally generalizes well to non-binary data with no further increase in complexity.

5.9.2.2 Correlated Data

Our algorithm relies on a fundamental assumption that $\vec{x}$ is a sequence of i.i.d. binary digits. Real world data such as audio and images often exhibit correlation in time or space. However, since the data is now correlated, our analysis which assumes that n_m (in Section 5.2) is the inner product of two i.i.d. vectors is not valid anymore. In this section, we consider a correlated signal model and propose suitable changes to the algorithm to account for this effect. More specifically, we introduce

a pre-processing step for both database and query, where we pass them through a whitening filter, which improves SNR of the correlation vector and thereby improving the sample and computational gain.

We consider the following signal model. We assume that the samples of the database $\{x[i]\}$ are from a (weakly) stationary time series (or, can be processed to approximate a stationary series by removing trend, seasonality etc.) generated by a ARMA(p, q) process given by

$$\tilde{x}_t = c + \varepsilon_t + \sum_{i=1}^{p} \phi_i \tilde{x}_{t-i} + \sum_{j=1}^{q} \theta_i \varepsilon_{t-i}$$

where $\tilde{x}_t = x[t] - \bar{x}$, with $\bar{x}$ referring to the mean of the stationary process $x[t]$, c is a constant, and $\varepsilon_t,\ \varepsilon_{t-1}, \cdots, \varepsilon_{t-q}$ are white noise error terms.

Notice that a valid query (that exists in the database) $\vec{y}$ that needs to be searched also follows the same ARMA(p, q) process as database. The problem is to find the locations $\vec{\tau} = \{\tau_1, \tau_2, \cdots, \tau_L\}$ where the query matches exactly or approximately.

We solve this problem in a three step approach as described below. Figure.5.6 illustrates our approach.

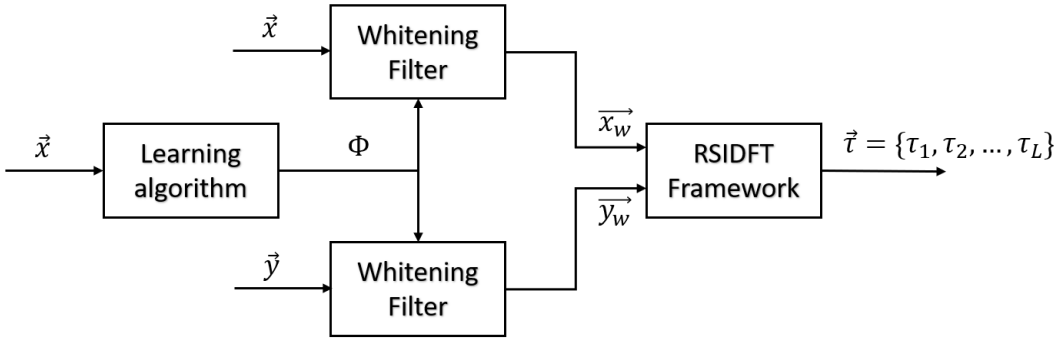

Figure 5.6: System model for non- i.i.d. pattern matching.

1. *Learning algorithm:* Learn the parameters $\Phi = \{\phi_1, \phi_2, \cdots, \phi_p,\ \theta_1, \cdots, \theta_q\}$ of the ARMA(p, q) process that approximates the database.

2. *Whitening filter:* Construct a whitening filter using the parameters Φ learnt by the learning algorithm. The database and query are whitened (uncorrelated) to obtain $\vec{\mathrm{x}}_w$ and $\vec{\mathrm{y}}_w$ respectively.

3. *RSIDFT framework:* Perform pattern matching on $\vec{\mathrm{x}}_w$, $\vec{\mathrm{y}}_w$ using the RSIDFT framework to identify the locations $\vec{\tau}$

Simulation Results: Simulations were performed to measure gains from the whitening operation. We first simulate a correlated signal $\vec{\mathrm{x}}_{corr}$ of length $N = 10^9$ by applying a Moving Average filter with coefficients $[0.45, 0.45, 0.45, 0.45, 0.45]$ to a signal with its elements sampled from $\mathcal{N}(0, 1)$. A whitened signal $\vec{\mathrm{x}}_{white}$ was generated by whitening $\vec{\mathrm{x}}_{corr}$ with the inverse filter. Pattern matching was performed on $\vec{\mathrm{x}}_{corr}$ and $\vec{\mathrm{x}}_{white}$ and their sample gains, defined as the ratio of N to the number of samples used by the sketch of database, G_{corr} and G_{white} were measured for two query lengths $M = 10^3$ (short query) and $M = 10^5$ (long query) and the results are shown in the Table 5.3. In both the cases, there was $3\times$ gain G_{white}/G_{corr} in sample complexity of $\vec{\mathrm{x}}_{white}$ compared to $\vec{\mathrm{x}}_{corr}$.

Note: The scaling factor of 0.45 was chosen to maintain same empirical mean and variance for $\vec{\mathrm{x}}_{corr}$ and $\vec{\mathrm{x}}_{white}$.

M	μ	G_{corr}	G_{white}	G_{white}/G_{corr}
10^3	0.33	$3.99\times$	$12.6\times$	$3.15\times$
10^5	0.56	$277.7\times$	$865.6\times$	$3.11\times$

Table 5.3: Comparison of sample gains for correlated and whitened signals.

5.9.3 Secure Pattern Matching

Mining large-scale data sets for information is the enabler to making informed decisions in every aspect of modern life from health care to finance. However, access to such pertinent information presents serious privacy and security concerns and balancing the utility of data with privacy concerns is a critical issue. This is particularly relevant when organizations that own the data and clients who want to query the data communicate with each other through a third party (outsourced) organization which provides software and computation as a service. In such a scenario, the organization holding the data may not want to reveal the identity of the entire data to the third party due to security and privacy concerns. Similarly, the client may not want to reveal the identity of the query entirely, but wants to reveal just enough information to see if the query is contained in the data set. Two examples provided in [41] exemplify this situation. As pointed out in [42], an airline that has a passenger list may not want to reveal all the passenger names whereas a government agency may want to check if a suspect was on the list. A health care provider holding a genome sequence of individual patients may not want to reveal the entire data set to a third party, but may want to reveal enough information to answer if certain portions of the genome sequence match DNA sequences of patients [43].

Secure pattern matching has been studied in the computer science literature and a summary of the techniques is given in [41]. No single technique appears to be optimal in terms of sketching, querying complexities and in providing privacy simultaneously. In fact, all the algorithm discussed there have at least linear complexity. The RSIDFT algorithm naturally has desirable privacy preserving features. We propose the following scheme - both the owner/sender and the client send only the samples of the Fourier transform to the third party, which runs the RSIDFT algo-

rithm to determine if there is a match or not. In this case, the third party does not have enough information to determine the identity of either $\vec{x}$ or $\vec{y}$, yet can answer the query correctly. Here, we are exploiting the fact neither $\vec{x}$ or $\vec{y}$ is sparse, but the correlation $\vec{r}$ is sparse. Notice that in the pattern matching problem, when the query is answered, the location of the match is determined. In a variation of this problem, we can ask for the query to be answered only by a yes or no without revealing information about the location of the match. In this case, the sample and computational complexities of the algorithm can be reduced even further. Specifically, we can completely avoid the second step in the decoding algorithm in Section 5.4.1 since we only want to determine if any bin is a singleton or multi-ton. In this case, the complexity can be reduced further. Following section describes the system model in detail.

System Model

The system involves three parties. Figure 5.7 provides an illustration of the system model.

- *Sender (SEN):* The sender SEN owns the actual database $\vec{x}$ of length N. SEN is assumed to have limited computational resources and hence hands off a private key (key) encrypted database $f_e(\vec{x}, key)$ to a third-party server SER to perform pattern matching.

- *Client (CLI):* The client holds the query (substring) $\vec{y}$ and wants to know the locations of match in the database $\vec{x}$. To accomplish this, CLI sends an encrypted version (using the same key generated by sender) of the query $f_e(\vec{y}, key)$ to SER and gets back the matching locations $\vec{\tau} = (\tau_1, \tau_2, \cdots, \tau_L)$.

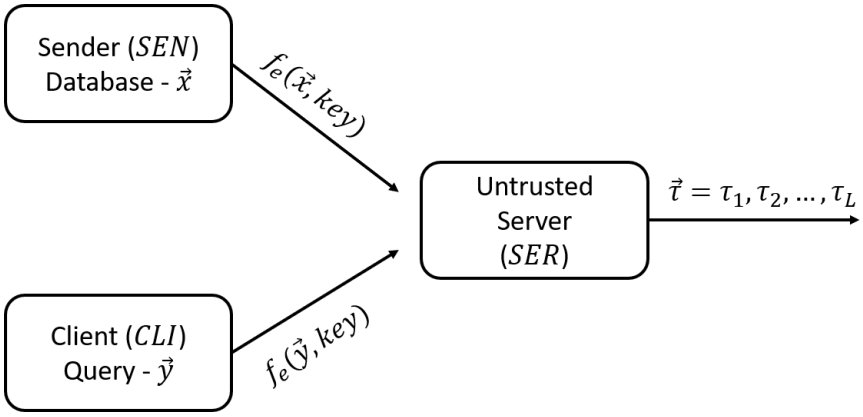

Figure 5.7: System model for outsourced pattern matching.

- *Third-party server (SER):* This server is assumed to have enough computational power to perform the pattern matching operation to identify the locations where a query is present in the database. Moreover, SEN and CLI do not trust SER and only provide an encoded version (with no access to key) of database and query. The objective of the SER is to estimate the locations $\vec{\tau}$ by just operating on $f_e(\vec{\mathrm{x}}, key)$ and $f_e(\vec{\mathrm{y}}, key)$. The third-party server performs the same functions as a RSIDFT framework in Section 5.4.1, with input $R = f_e(\vec{\mathrm{x}}, key) \odot f_e(\vec{\mathrm{y}}, key)$.

We consider an outsourced pattern matching model, where the party holding the database (called sender SEN) is computationally weak, and wishes to outsource the data and computations to an untrusted third party server (SER). Given the untrusted behavior of the server, we need to protect privacy of the database $\vec{\mathrm{x}}$ and query $\vec{\mathrm{y}}$, and yet want the server to compute all locations of matches with high probability. We also want to minimize the communication cost, time and space complexity of the algorithm.

Note that the encryption function f_e should preserve the privacy of $\vec{\mathrm{x}}$ and $\vec{\mathrm{y}}$ and only provide enough information to recover the locations $\vec{\tau}$ and not learn anything more about the query or database. The design of encoding function is described in detail in Section 5.9.3.

Encryption function

We propose two separate encryption functions for i.i.d. and non-i.i.d. (correlated) databases.

1. **i.i.d. database:** Consider the random setting in which the database samples $x[i]$'s form a sequence of independent and identically distributed (i.i.d.) random variables, each taking values in $A \subset \{+1, -1\}$. Figure 5.8 illustrates the encoding operation for both sender and client. The encoding operation involves two steps - sketching and encryption. The sketching operation is same as described in Section 5.4.1, where we take $|S|$ samples $\vec{X}[S]$ from the Fourier coefficients of $\vec{\mathrm{x}}$ based on index set S induced by the RSIDFT architecture. The encryption operation, for a given $key \in \{+1, -1\}^{|S|}$ is defined by

$$f_e(\vec{\mathrm{x}}, key) = \vec{X}[S] \odot key$$
$$f_e(\vec{\mathrm{y}}, key) = \vec{Y}'[S] \odot key$$

   where, $\odot$ is element-wise multiplication operation. It can be seen that, if the secret key is chosen at random, the untrusted server (SER) receives only a sign garbled version of the sub-sampled Fourier coefficients, which does not provide enough information to learn about the actual message $\vec{\mathrm{x}}$ or $\vec{\mathrm{y}}$.

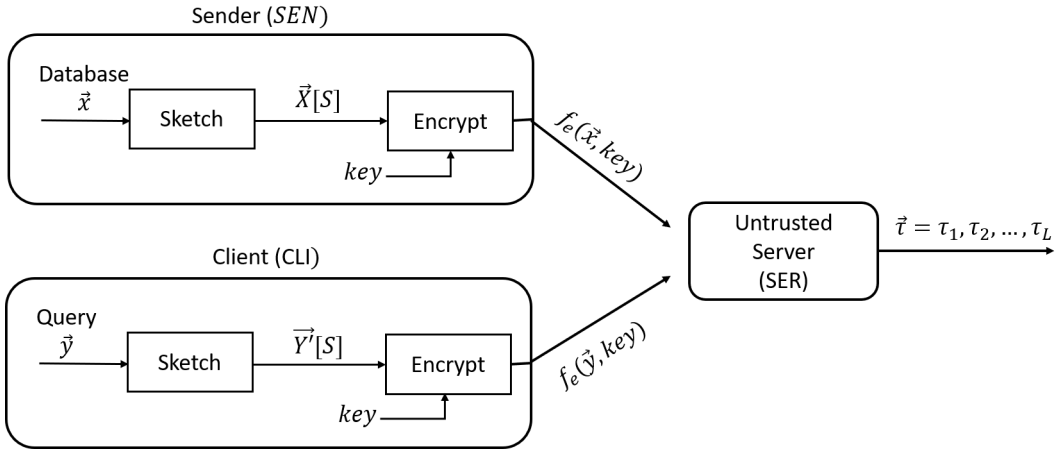

Figure 5.8: Encryption for Secured Pattern Matching with i.i.d. database.

2. **Non-i.i.d. database:**

   Consider the case when the database samples $x[i]$'s form a sequence of correlated random variables, according to a stationary process $ARMA(p, q)$ defined in Section 5.9.2.2, each taking values in $\mathbb{R}$. Figure 5.9 illustrates the encoding operation for both sender and client. The encoding operation involves two steps - whitening and sketching. Let $\vec{\Phi} = \{\phi_1, \phi_2, \cdots, \phi_p,\ \theta_1, \cdots, \theta_q\}$ denote the parameters of the whitening filter, and $\vec{\mathrm{x}}_w$, $\vec{\mathrm{y}}_w$ denote the output of the whitening filter with $\vec{\mathrm{x}}$ and $\vec{\mathrm{y}}$ as inputs respectively. The sketching operation is same as described in Section 5.4.1, where we take $|S|$ samples $\vec{X}_w[S]$ (or, $\vec{Y}'_w[S]$) from the Fourier coefficients of $\vec{\mathrm{x}}_w$ (or, $\vec{\mathrm{y}}_w$) based on index set S induced by the RSIDFT architecture. Hence the encrypted signals of $\vec{\mathrm{x}}$ and $\vec{\mathrm{y}}$ are $f_e(\vec{\mathrm{x}}, key) = \vec{X}_w[S]$ and $f_e(\vec{\mathrm{y}}, key) = \vec{Y}'_w[S]$ respectively.

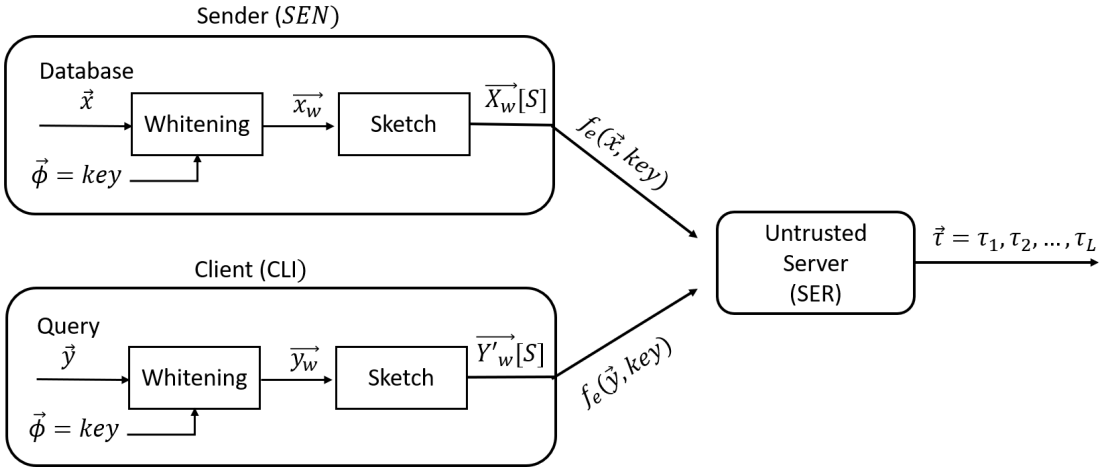

Figure 5.9: Encryption for Secured Pattern Matching with non- i.i.d. database.

Notice that the secret $key = \vec{\Phi}$ is chosen to uncorrelate/whiten the signal and hence the untrusted server (SER) receives only a sub-sampled version of the Fourier coefficients of a random i.i.d vector, which does not give enough information to learn about the actual message $\vec{x}$ or $\vec{y}$. Unlike the i.i.d case, the key can be distilled from database $\vec{x}$ and query $\vec{y}$, and hence, there is no need for key sharing.

6. CONCLUSIONS

In this dissertation we focused on three sparse recovery problems i) computing a sparse Fourier transform, ii) computing a sparse Hadamard transform and iii) pattern matching. We leveraged connections between coding theory and sub-linear time sparse recovery problems like sparse Fourier and Hadamard transform computations. This relation allowed us to understand the properties of established recovery algorithms under different signal models and to suggest algorithmic enhancements to minimize space and time complexity.

We started by illustrating the relationship between an extended Fast Fourier Aliasing-based Sparse Transform (FFAST) algorithm and the iterative hard decision decoding of product codes. We have seen that the FFAST algorithm is analogous to an iterative decoder for a carefully defined product code, thresholds for which can be determined by an extension of Justensen [1] analysis to d dimensional product codes. Interpreting the FFAST algorithm as decoding of a product code also provided insights into the performance of the FFAST algorithm when non-zero coefficients are not chosen randonly, but are bursty, as encountered in many practical applications like spectrum sensing. It was further observed that the FFAST algorithm performs better for bursty signals in comparison to those for randomly chosen non-zero coefficients.

We then considered the problem of computing the Walsh-Hadamard Transform (WHT) of an $N = 2^n$-dimensional signal whose WHT is K-sparse, when the sparsity parameter $K = O(N^\delta)$ scales sub-linearly in N for some $0 < \delta < 1$. Using lessons from the earlier FFAST algorithm analysis, we proposed improvements to the algorithm in [2]. Furthermore, through density evolution analysis and simulations we

showed that the proposed modification substantially improves the algorithm's space and time complexity, often achieving as much as a 70% reduction.

We concluded by considering the substring/pattern matching problem of querying a string (or a database) of length N bits to determine all the locations where a substring (query) of length M appears either exactly or is within a Hamming distance of K from the query. We proposed algorithms for Exact Pattern Matching problem where M consecutive symbols from $\vec{x}$ and is presented as a query, and the Approximate Pattern Matching problem where we assume a noisy version of a substring, and evaluated based on the sketching complexity, and the computational complexity in answering the query. Using a sparse Fourier transform computation based approach we showed that all such matches can be determined with high probability in sub-linear time and space. Further, several extensions of the main theme were also discussed.

REFERENCES

[1] J. Justesen and T. Høholdt, "Analysis of iterated hard decision decoding of product codes with reed-solomon component codes," in *Information Theory Workshop, 2007. ITW'07. IEEE*, pp. 174–177, IEEE, 2007.

[2] R. Scheibler, S. Haghighatshoar, and M. Vetterli, "A fast hadamard transform for signals with sublinear sparsity in the transform domain," *IEEE Transactions on Information Theory*, vol. 61, no. 4, pp. 2115–2132, 2015.

[3] S. Pawar and K. Ramchandran, "Computing a k-sparse n-length discrete fourier transform using at most $4k$ samples and $o(k \log k)$ complexity," pp. 464–468, IEEE, 2013.

[4] S. Muthukrishnan, *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

[5] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 234–247, 2003.

[6] P. Indyk, "Sketching, streaming and sublinear-space algorithms," *Graduate course notes, available at*, vol. 33, p. 617, 2007.

[7] N. Shental, A. Amir, and O. Zuk, "Rare-allele detection using compressed se (que) nsing," *arXiv preprint arXiv:0909.0400*, 2009.

[8] Y. Erlich, N. Shental, A. Amir, and O. Zuk, "Compressed sensing approach for high throughput carrier screen," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 539–544, IEEE, 2009.

[9] R. M. Kainkaryam, A. Bruex, A. C. Gilbert, J. Schiefelbein, and P. J. Woolf, "poolmc: Smart pooling of mrna samples in microarray experiments," *BMC bioinformatics*, vol. 11, no. 1, p. 299, 2010.

[10] T. Tao and V. H. Vu, *Additive combinatorics*, vol. 105. Cambridge University Press, 2006.

[11] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid mr imaging," *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[12] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[13] R. Clifford, K. Efremenko, E. Porat, and A. Rothschild, "k-mismatch with don't cares," in *European Symposium on Algorithms*, pp. 151–162, Springer, 2007.

[14] A. Andoni, H. Hassanieh, P. Indyk, and D. Katabi, "Shift finding in sub-linear time," in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 457–465, Society for Industrial and Applied Mathematics, 2013.

[15] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Near-optimal algorithm for sparse fourier transform," 2012.

[16] A. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse fourier transform: A compressed fourier transform for big data," vol. 31, no. 5, pp. 91–100, 2014.

[17] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast fourier sampling," vol. 25, no. 2, pp. 57–66, 2008.

[18] T. Richardson and R. Urbanke, *Modern coding theory*. Cambridge University Press, 2008.

[19] M. Luby, "LT codes," pp. 271–271, 2002.

[20] S. Pawar and K. Ramchandran, "A robust R-FFAST framework for computing a k-sparse n-length DFT in $\mathrm{O}(k \log\ n)$ sample complexity using sparse-graph codes," pp. 1852–1856, 2014.

[21] N. T. Janakiraman, S. Emmadi, K. Narayanan, and K. Ramchandran, "Exploring connections between sparse fourier transform computation and decoding of product codes," in *53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 1366–1373, 2015.

[22] S. Pawar and K. Ramchandran, "A FFAST framework for computing a k-sparse dft in $o(klogk)$ time using sparse-graph alias codes." http://arxiv.org/pdf/1305.0870.pdf, 2013.

[23] J. Justesen, "Performance of product codes and related structures with iterated decoding," vol. 59, no. 2, pp. 407–415, 2011.

[24] H. Burton and J. Weldon, E., "Cyclic product codes," *Information Theory, IEEE Transactions on*, vol. 11, pp. 433–439, Jul 1965.

[25] Y.-Y. Jian, H. D. Pfister, and K. R. Narayanan, "Approaching capacity at high rates with iterative hard-decision decoding," pp. 2696–2700, 2012.

[26] K. J. Horadam, *Hadamard matrices and their applications*. Princeton university press, 2012.

[27] M. Cheraghchi and P. Indyk, "Nearly optimal deterministic algorithm for sparse Walsh-Hadamard transform," *ACM Transactions on Algorithms (TALG)*, vol. 13, no. 3, p. 34, 2017.

[28] X. Li, J. K. Bradley, S. Pawar, and K. Ramchandran, "The spright algorithm for robust sparse hadamard transforms," in *IEEE Int. Symp. Info. Theory*, pp. 1857–1861, 2014.

[29] A. Amir, M. Lewenstein, and E. Porat, "Faster algorithms for string matching with k mismatches," *Journal of Algorithms*, vol. 50, no. 2, pp. 257–275, 2004.

[30] G. Navarro, "A guided tour to approximate string matching," *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.

[31] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[32] P. Ferragina and G. Manzini, "Indexing compressed text," *Journal of the ACM (JACM)*, vol. 52, no. 4, pp. 552–581, 2005.

[33] H. Li and R. Durbin, "Fast and accurate short read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[34] H. Li and R. Durbin, "Fast and accurate long-read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, 2010.

[35] N. Zhang, A. Mukherjee, D. Adjeroh, and T. Bell, "Approximate pattern matching using the burrows-wheeler transform," in *Proceedings of the Conference on Data Compression*, p. 458, IEEE Computer Society, 2003.

[36] W. I. Chang and T. G. Marr, "Approximate string matching and local similarity," in *Annual Symposium on Combinatorial Pattern Matching*, pp. 259–273, Springer, 1994.

[37] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk, "Faster GPS via the sparse Fourier transform," in *mobicom*, pp. 353–364, ACM, 2012.

[38] K. Lee, R. Pedarsani, and K. Ramchandran, "Saffron: A fast, efficient, and robust framework for group testing based on sparse-graph codes," *arXiv preprint arXiv:1508.04485*, 2015.

[39] S. Kay, "A fast and accurate single frequency estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1987–1990, 1989.

[40] X. Li, S. Pawar, and K. Ramchandran, "Sub-linear time compressed sensing using sparse-graph codes." `http://www.eecs.berkeley.edu/~xiaoli/TR_CS_sublinear.pdf`.

[41] K. El Defrawy and S. Faber, "Blindfolded data search via secure pattern matching," *Computer*, vol. 46, no. 12, pp. 68–75, 2013.

[42] W. J. Krouse and B. Elias, "Terrorist watchlist checks and air passenger prescreening," in *"Congressional Research Service, 30 Dec. 2009; www.fas.org/sgp/crs/homesec/RL33645.pdf"*, DTIC Document, 2009.

[43] J. Katz and L. Malka, "Secure text processing with applications to private DNA matching," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 485–492, ACM, 2010.